Funktionsbeschreibung Ausgabe 11/2002

# sinumerik

Synchronaktionen

SINUMERIK 840D/840Di/810D

SIEMENS

SIEMENS

# SINUMERIK 840D/840Di/810D

# Synchronaktionen

## Funktionsbeschreibung



**Gültig für**

| *Steuerung* | *Softwarestand* |
|---|---|
| SINUMERIK 840D | 6 |
| SINUMERIK 840DE (Exportversion) | 6 |
| SINUMERIK 840D powerline | 6 |
| SINUMERIK 840DE powerline | 6 |
| SINUMERIK 840Di | 2 |
| SINUMERIK 840DiE (Exportversion) | 2 |
| SINUMERIK 810D | 3 |
| SINUMERIK 810DE (Exportversion) | 3 |
| SINUMERIK 810D powerline | 6 |
| SINUMERIK 810DE powerline | 6 |

**Ausgabe 11.2002**

## SINUMERIK®–Dokumentation

### Auflagenschlüssel

Die nachfolgend aufgeführten Ausgaben sind bis zur vorliegenden Ausgabe erschienen.

In der Spalte ”Bemerkung” ist durch Buchstaben gekennzeichnet, welchen Status die bisher erschienen Ausgaben besitzen.

*Kennzeichnung des Status in der Spalte ”Bemerkung”:*

**A** . . . . . Neue Dokumentation.
**B** . . . . . Unveränderter Nachdruck mit neuer Bestell–Nummer
**C** . . . . . Überarbeitete Version mit neuem Ausgabestand.
Hat sich der auf der Seite dargestellte technische Sachverhalt gegenüber dem vorherigen Ausgabestand geändert, wird dies durch den veränderten Ausgabestand in der Kopfzeile der jeweiligen Seite angezeigt.

| | | |
|---|---|---|
| 06.94 | 6FC5 297–0AC30–0AP0 | **A** |
| 08.94 | 6FC5 297–0AC30–0AP1 | **C** |
| 02.95 | 6FC5 297–2AC30–0AP0 | **C** |
| 04.95 | 6FC5 297–2AC30–0AP1 | **C** |
| 09.95 | 6FC5 297–3AC30–0AP0 | **C** |
| 03.96 | 6FC5 297–3AC30–0AP1 | **C** |
| 08.97 | 6FC5 297–4AD40–0AP0 | **A**1) |
| 12.97 | 6FC5 297–4AD40–0AP1 | **C** |
| 12.98 | 6FC5 297–5AD40–0AP0 | **C** |
| 08.99 | 6FC5 297–5AD40–0AP1 | **C** |
| 04.00 | 6FC5 297–5AD40–0AP2 | **C** |
| 10.00 | 6FC5 297–6AD40–0AP0 | **C** |
| 09.01 | 6FC5 297–6AD40–0AP1 | **C** |
| 11.02 | 6FC5 297–6AD40–0AP2 | **C** |

1) Die vorliegende Dokumentation löst die Funktion S5 ab, welche für die SW–Stände davor in der Druckschrift ’Funktionsbeschreibung Erweiterungsfunktionen’ beschrieben wurde.

Dieses Buch ist Bestandteil der Dokumentation auf CD–ROM (**DOCONCD**)

| **Ausgabe** | **Bestell–Nr.** | **Bemerkung** |
|---|---|---|
| 11.02 | 6FC5 298–6CA00–0AG3 | **C** |

**Marken**
SIMATIC®, SIMATIC HMI®, SIMATIC NET®, SIROTEC®, SINUMERIK® und SIMODRIVE® sind einge–tragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Druckschrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Weitere Informationen finden Sie im Internet unter:
http://www.ad.siemens.de/sinumerik

Die Erstellung dieser Unterlage erfolgte mit Interleaf V 7



Es können weitere, in dieser Dokumentation nicht beschriebene Funktionen in der Steuerung lauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei Neulieferung bzw. im Servicefall.

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard– und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen vorbehalten.

Bestell–Nr. 6FC5 297–6AD40–0AP2
Printed in the Federal Republic of Germany

Siemens–Aktiengesellschaft

# Vorwort

## Lesehinweise

Die SINUMERIK–Dokumentation ist in 3 Ebenen gegliedert:

- Allgemeine Dokumentation
- Anwender–Dokumentation
- Hersteller/Service–Dokumentation

Die vorliegende Dokumentation wendet sich an den Werkzeugmaschinen–Hersteller. Die Druckschrift beschreibt ausführlich die in den Steuerungen SINUMERIK 840D/810D vorhandenen Funktionalitäten.

Die Funktionsbeschreibungen sind nur für den speziellen bzw. bis zum aufgeführten Softwarestand gültig. Bei neuen Softwareständen sind die dazu gültigen Funktionsbeschreibungen anzufordern. Alte Funktionsbeschreibungen sind für neue Softwarestände nur noch teilweise verwendbar.

Nähere Informationen zu weiteren Druckschriften über SINUMERIK 840D/840Di/810D sowie zu Druckschriften, die für alle SINUMERIK–Steuerungen gelten (z.B. Universalschnittstelle, Meßzyklen ...) erhalten Sie von Ihrer Siemens–Niederlassung.

---

**Hinweis**

Es können in der Steuerung weitere, in dieser Dokumentation nicht erläuterte, Funktionen ablauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei Neulieferung bzw. im Servicefall.

---

## Hotline

Bei Fragen zur Steuerung wenden Sie sich bitte an folgende Hotline:

A&D Technical Support
Tel.: +49 (180) 5050 222
Fax: +49 (180) 5050–223
E–Mail: adsupport@siemens.com

Bei Fragen zur Dokumentation (Anregungen, Korrekturen) senden Sie bitte ein Fax an folgende Faxadresse:
Fax: +49 (9131) 98–2176
E–Mail: motioncontrol.docu@erlf.siemens.de

Faxformular: siehe Rückmeldeblatt am Schluss der Druckschrift

## Internetadresse SINUMERIK

http://www.ad.siemens.de/sinumerik

**SINUMERIK 840D powerline**

Ab 09.2001 steht die

- SINUMERIK 840D powerline und
- SINUMERIK 840DE powerline

mit verbesserter Performance zur Verfügung. Eine Auflistung der verfügbaren **powerline**–Baugruppen finden Sie in der Hardware–Beschreibung /PHD/ in Kapitel 1.1.

**SINUMERIK 810D powerline**

Ab 12.2001 steht die

- SINUMERIK 810D powerline und
- SINUMERIK  810DE powerline

mit verbesserter Performance zur Verfügung. Eine Auflistung der verfügbaren **powerline**–Baugruppen finden Sie in der Hardware–Beschreibung /PHC/ in Kapitel 1.1.

**Zielsetzung**

Die vorliegende Dokumentation beschreibt die Funktion Synchronaktionen für SINUMERIK 840D ab SW 4 und für SINUMERIK 810D ab SW 2. Sie löst die Funktion S5 ab, welche für die SW–Stände davor
in der Druckschrift 'Funktionsbeschreibung Erweiterungsfunktionen' beschrieben wurde.

Die Funktionsbeschreibungen vermitteln die für die Projektierung und Inbetriebnahme benötigten Informationen.

**Zielgruppen**

Die Funktionsbeschreibungen beinhalten damit Informationen für:

- den Projekteur der Anlage
- den PLC–Programmierer bei der Erstellung des PLC–Anwenderprogramms mit den aufgelisteten Signalen
- den Inbetriebnehmer nach der Projektierung und Aufbau der Anlage
- den Servicetechniker zur Überprüfung und Interpretierung der Statusanzeigen und Alarme

---

!

**Wichtig**

Diese Dokumentation ist gültig für:

- Steuerung SINUMERIK 840D,
  Softwarestand 6
- Steuerung SINUMERIK 810D,
  Softwarestand 6
- Steuerung SINUMERIK 840Di
  Softwarestand 2

---

**Angabe des SW–Standes**

Die in der Dokumentation angegebenen SW–Stände beziehen sich auf die Steuerung SINUMERIK 840D, der dazu parallel gültige SW–Stand für die Steuerung SINUMERIK 810D (falls die Funktion frei gegeben ist, siehe /BU/, Katalog NC 60) wird nicht jeweils explizit angegeben. Dabei gilt:

Tabelle 1-1 Entsprechung des SW–Standes

| SINUMERIK 840D | | SINUMERIK 810D | SINUMERIK 810D powerline | SINUMERIK 840Di |
|---|---|---|---|---|
| 6.3 (09.01) | entspricht | – | 6.1 (12.01) | 2.1 (07.01) |
| 4.3 (04.00) | entspricht | 2.3 (12.97) | – | 1.1 (07.00) |
| 3.7 (03.97) | entspricht | 1.7 (03.97) | – | – |

## Symbole



**Gefahr**

Dieser Warnhinweis bedeutet, daß Tod, schwere Körperverletzung oder erheblicher Sachschaden eintreten **werden**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.



**Warnung**

Dieser Warnhinweis bedeutet, daß Tod, schwere Körperverletzung oder erheblicher Sachschaden eintreten **können**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.



**Vorsicht**

Dieser Warnhinweis (mit Warndreieck) bedeutet, daß eine leichte Körperverletzung oder ein Sachschaden eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

**Vorsicht**

Dieser Warnhinweis (ohne Warndreieck) bedeutet, daß ein Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

**Achtung**

Dieser Warnhinweis bedeutet, daß ein unerwünschtes Ereignis eintreten **kann**, wenn der entsprechende Hinweis nicht beachtet wird.

!

**Wichtig**

Dieser Hinweis bedeutet, daß ein wichtiger Sachverhalt zu beachten ist.

**Hinweis**

Dieser Hinweis bedeutet, daß auf einen weiteren Sachverhalt hingewiesen wird.

# Inhalt

■

# Platz für Notizen

# Kurzbeschreibung 1

**Definition Synchronaktionen**

Bewegungssynchronaktionen (kurz Synchronaktionen) sind vom Anwender programmierte Anweisungen, die synchron zur Bearbeitung des Teileprogrammes im Interpolationstakt vom NCK ausgewertet werden. Ist die in der Synchronaktion enthaltene Bedingung erfüllt oder keine angegeben, so werden zugeordnete Aktionen synchron zur weiteren Bearbeitung aktiviert.

**Anwendungen**

Der folgende Auszug aus den vielfältigen Möglichkeiten gibt Hinweise zu den Einsatzmöglichkeiten für Aktionen in Synchronaktionen.

- Ausgabe von Hilfsfunktionen an die PLC
- Schreiben und Lesen von Echtzeitvariablen
- Positionierung von Achsen–Spindeln
- Aktivierung von Synchronprozeduren wie z.B.:
  - Einlesesperre
  - Restweglöschen
  - Vorlaufstop beenden
- Aktivierung von Technologiezyklen
- Online–Berechnung von Funktionswerten
- Online–Werkzeugkorrekturen
- Kopplungen/Mitschleppen aktivieren/deaktivieren
- Messungen ausführen
- Sperren–Freigeben von Synchronaktionen

In Kapitel ”Ausführliche Beschreibung” werden alle Möglichkeiten dargestellt.

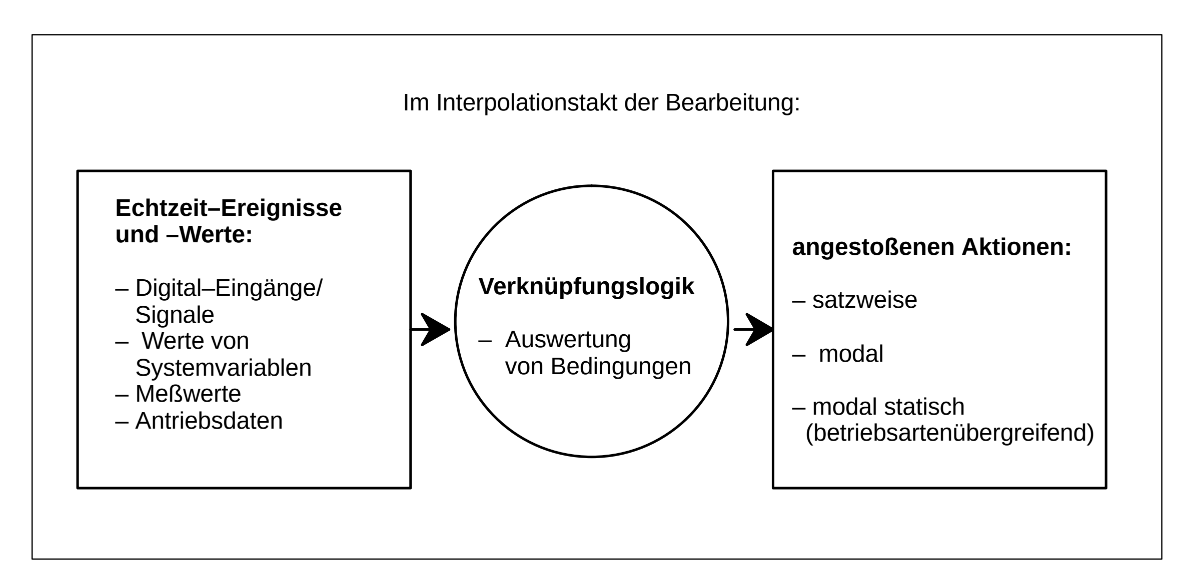



Bild 1-1 Synchronaktionen schematisch

Die Details der Programmierung von Synchronaktionen finden Sie in

**Literatur:** /PGA/, Programmieranleitung Arbeitsvorbereitung

Die folgenden Kapitel beschreiben die:

– funktionalen Zusammenhänge für Synchronaktionen in Kapitel 2,

– sowie die erforderlichen Maschinendaten in Kapitel 4,

– Anwendungsbeispiele in Kapitel 6.

---

**Hinweis**

Die vorliegende Beschreibung umfaßt die Funktionalität des Software–Standes 5. Die Leistungen der Synchronaktionen bis einschließlich SW–Stand 3 sind beschrieben in:
**Literatur:** /FB/, S5, ”Synchronaktionen”

---

■

# Ausführliche Beschreibung 2

## 2.1 Komponenten von Synchronaktionen

**Struktur einer Synchronaktion**

| **Kompo-nente:** | **Gültigkeit, Identifikati-onsnummer** | **Häufigkeit** | **G–Code für Bed. und Aktion** | **Bedingung** | **Aktions-kennwort** (fest) | **G–Code für Aktion** | **Aktion oder Technolo-gie–zyklus** S. 2.5 |
|---|---|---|---|---|---|---|---|
| Beispiel: | IDS=1 | EVERY | G70 | $AAA_IM[B] > 15 | DO | G71 | POS[X]= 100 |

Die Bestandteile der Synchronaktion:

- Gültigkeit:
  - mit Identifikationsnummer
  - ohne Identifikationsnummer
- Häufigkeit
- G–Code für Bedingung und Aktion (ab SW 5)
- Bedingung
- G–Code für Aktionen (ab SW 5)
- Aktion(en) / Technologiezyklus

werden im Folgenden einzeln erklärt.

**Gültigkeit, ID–Nummer**

Für die Festlegung des Gültigkeitsbereiches einer Synchronaktion bieten sich drei Möglichkeiten:

- keine Angabe
- ID
- IDS

Keine Angabe

Synchronaktionen ohne Gültigkeitsangabe wirken satzweise, d.h. nur für den darauffolgenden Satz.

Satzweise wirksame Synchronaktionen wirken nur im **AUTOMATIK**–Betrieb.

Satzweise wirksame Synchronaktionen wirken **ab SW 6.1** modal über alle Vorlauf–Stop–Sätze (auch implizit erzeugte) und über implizit erzeugte Zwischensätze.

ID

Synchronaktionen mit Gültigkeitskennung **ID** wirken **modal** in darauffolgend programmierten Sätzen. Sie wirken nur im **AUTOMATIK**–Betrieb.

Begrenzung:

– bis eine andere Synchronaktion mit gleicher Identifikationsnummer programmiert wird

– bis zum Löschen, CANCEL(i), S. Kap. 2.5.1.

IDS

Statisch wirksame Synchronaktionen, die mit dem Schlüsselwort "**IDS**" programmiert werden, sind **in allen Betriebsarten** aktiv. Sie werden auch als statische Synchronaktionen bezeichnet. Option.

Das Löschen von Synchronaktionen mit ID oder IDS erfolgt aus dem Teileprogramm heraus.

**Identifikations-nummern**

Für modale Synchronaktionen (ID, IDS) werden Identifikationsnummern zwischen 1 und 255 vergeben. Sie sind für die Funktionen der gegenseitigen Koordinierung von Synchronaktionen von Bedeutung. S. Kap. 2.5.1. Modale / statische Synchronaktionen mit Identifikationsnummern von 1 – 64 können von PLC aus verriegelt und freigegeben werden. S. 2.6.1.

Die Identifikationsnummern müssen im Kanal eindeutig vergeben werden.

**Anwendung für statische Synchronaktionen:**
– AC–Schleifen auch in Betriebsart JOG aktiv
– Verknüpfungslogik für Safety Integrated
– Überwachungsfunktionen, Reaktion auf Maschinenzustände in allen Betriebsarten
– Optimierung des Werkzeugwechsels
– Zyklische Maschinen

Beispiele:

```
IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100     alle Betriebsarten
ID=2  EVERY $A_IN[1]==0 DO POS[X]=0       AUTOMATIK
```

---

**Hinweis**

Folgende Aktionen sind nur in der Betriebsart AUTOMATIC bei aktivem Programm wirksam:
STOPREOF,
DELDTG

---

**Häufigkeit**

Durch das Schlüsselwörter (s. Tabelle) wird angegeben, wie oft die darauffolgende Bedingung abgefragt und die zugehörende Aktion bei erfüllter Bedingung ausgeführt werden soll. Die angegebenen Schlüsselworte sind Bestandteil der Synchronaktionsbedingung.

Tabelle 2-1 Wirkung der Häufigkeitsschlüsselwörter

| Schlüsselwort | Abfrage–Häufigkeit |
|---|---|
| keins | Ist keine Häufigkeitsangabe programmiert, so wird die Aktion **zyklisch** in jedem Interpolationstakt ausgeführt. |
| **WHENEVER** | Die zugehörigen Aktion/Technologiezyklus wird zyklisch in jedem Interpolationstakt ausgeführt, solange die **Bedingung erfüllt** ist. |
| **FROM** | Wenn die **Bedingung einmal** erfüllt ist, wird die Aktion/Technologiezyklus **zyklisch** in jedem Interpolationstakt ausgeführt, solange die Synchronaktion aktiv ist. |
| **WHEN** | Wenn die **Bedingung erfüllt** ist, wird die Aktion/Technologiezyklus **ein einziges Mal** ausgeführt. Ist die Aktion einmal ausgeführt, so wird die Bedingung nicht mehr überprüft. |
| **EVERY** | Die Aktion/Technologiezyklus wird **einmal** angestoßen, wenn die Bedingung erfüllt ist. Die Aktion/Technologiezyklus wird wieder ausgeführt, wenn die **Bedingung vom Zustand falsch in den Zustand wahr** übergeht.<br>Im Gegensatz zum Schlüsselwort WHEN bleibt die Überprüfung der Bedingung auch nach Ausführung der Aktion aktiv solange, bis die Synchronaktion gelöscht oder inaktiv geschaltet wird. |

Details zu Technologiezyklen finden Sie unter 2.5.

**Löschen**

Wird eine aktive Synchronaktion mit **CANCEL** aus dem Teileprogramm abgewählt (gelöscht), so wird die aktive Aktion nicht beeinflußt. Positionierbewegungen werden wie programmiert beendet. Mit dem Befehl CANCEL kann eine modal oder statisch wirksame Synchronaktion gelöscht werden.
Wird eine Synchronaktion gelöscht, während die daraus aktivierte Positionierachsbewegung noch aktiv ist, so wird die Positionierachsbewegung abgeschlossen. Ein Kanal–Stop bricht auch die Positionierbewegung aus Synchronaktionen/Technologiezyklen ab.

**G–Code für Bedingung und Aktion**

Ab **SW–Stand 5** sind G–Codes in Synchronaktionen programmierbar. Damit kann erreicht werden, daß unabhängig vom gerade aktiven Teileprogrammzustand für die Auswertung der Bedingung und die auszuführende Aktion/Technologiezyklus definierte Einstellungen bestehen. Die Abkopplung der Synchronaktionen vom Programmumfeld ist erforderlich, weil Synchronaktionen zu beliebigen Zeitpunkten aufgrund erfüllter Auslösebedingungen ihre Aktionen in definiertem Ausgangszustand ausführen sollen.

Anwendungsfälle:
Festlegung der Maßsysteme für Bedingungsauswertung und Aktion durch G–Codes G70, G71, G700, G710.

---

**Hinweis**

Bei **SW–Stand 5** ist die Verwendung der G–Codes in Synchronaktionen auf diese 4 G–Codes beschränkt.

---

Ein angegebener G–Code bei der Bedingung gilt für die Auswertung der Bedingung **und** für die Aktion, wenn bei der Aktion kein eigener G–Code angegeben ist.

Pro Bedingungssteil darf nur ein G–Code der G–Code–Gruppe programmiert werden.

**Bedingungen**

Die Ausführung der Aktionen/Technologiezyklen kann von einer Bedingung (**logischer Ausdruck**) abhängig gemacht werden.

Die Bedingung wird im Interpolationstakt überprüft. Ist keine Bedingung angegeben, so wird die Aktion zyklisch in jedem IPO–Takt ausgeführt.
Bis Softwarestand 3 ist als Bedingung der Vergleich einer Echtzeitvariable mit einem im Vorlauf berechneten Ausdruck oder der Vergleich zweier Echtzeitvariablen zulässig.
Beispiele:
WHENEVER **$AA_IM[X] > 10.5*SIN(45**) DO .... oder
WHENEVER **$AA_IM[X] > $$AA_IM[X1]** DO ...

Ab Softwarestand 4 ist es zusätzlich möglich, Vergleiche durch boolesche Verknüpfungen miteinander zu verbinden. Zulässig sind die booleschen Operatoren der NC–Sprache:
**NOT, AND**, **OR**, **XOR, B_OR, B_AND, B_XOR, B_NOT.**

Beispiele:
WHENEVER ($A_IN[1]==1) **OR** ($A_IN[3]==0) DO ...
; solange Eingang 1 ansteht oder Eingang 3 nicht ansteht ...

Innerhalb einer Bedingung können zwei oder mehr Echtzeit–Ausdrücke miteinander verglichen werden.
Vergleiche sind jeweils möglich zwischen **typgleichen** Variablen oder Teilausdrücken.
Beispiel:
WHEN $AA_IM[X2] <= **$AA_IM[X1] +.5** DO $AA_OVR[X1]=0
; Anhalten, wenn Sicherheitsabstand überschritten ist

Die Möglichkeiten des Echtzeitausdrucks sind im Kapitel ”Berechnungen in Echtzeit” beschrieben. Bei Bedingungen können alle die Systemvariablen angesprochen werden, die in Kapitel 2.3.8 angegeben sind. Darüberhinaus:

- Maschinendaten z.B. $$MN_..., $$MC_..., $$MA_...
- Settingdaten z.B. $$SN_..., $$SC_..., $$SA_...

---

**Hinweis**

- GUD–Variablen können nicht verwendet werden
- R–Parameter werden mit $R... adressiert.
- Settingdaten und Maschinendaten, deren Wert sich während der Bearbeitung ändern kann, müssen mit **$$**S._... / **$$**M._... programmiert werden.

---

Weitere Beispiele zu Bedingungen finden Sie in 6.1.

**G–Code für die Aktion**

Dieser G–Code gibt für alle Aktionen im Satz und Technologiezyklen ggf. einen anderen G–Code als den bei der Bedingung gesetzten vor. Sind Technologiezyklen im Aktionsteil, so gilt der G–Code auch nach Abschluß des Technologiezyklus' für alle darauffolgenden Aktionen bis zum nächsten G–Code modal weiter. Pro Aktionsteil darf nur ein G–Code der G–Code–Gruppe programmiert werden.

**Aktionen**

In jeder Synchronaktion werden ein oder mehrere Aktionen oder ein Technologiezyklus programmiert. Diese werden ausgeführt, wenn die Bedingung erfüllt ist. Sind mehrere Aktionen in einer Synchronaktion programmiert, so werden diese im selben Interpolationstakt ausgeführt.

Beispiel: WHEN $AA_IM[Y] >= 35.7 DO **M135 $A_OUT[1]=1**
Wenn der Istwert der Y–Achse größer oder gleich 35.7 ist, dann wird M135 an PLC ausgegeben und gleichzeitig der Ausgang 1 gesetzt.

Programm/
Technologiezyklus

Als Aktion kann auch ein Programm (Name) angegeben werden. In diesem Programm sind alle Aktionen zulässig, die einzeln in Synchronaktionen programmiert werden können. Diese Programme werden im Folgenden auch als **Technologiezyklen** bezeichnet. Ein Technologiezyklus ist eine Folge von Aktionen, die sequentiell im Interpolationstakt abgearbeitet werden. S. 2.5.

Anwendung: Einzelachsprogramme, Zyklische Maschinen.

**Bearbeitungsvorgang**

Die Sätze eines Teileprogramms werden in der Programmaufbereitung vorbereitet, abgespeichert und dann sequentiell in der Interpolationsebene (Hauptlauf) abgearbeitet. Der Zugriff auf Variable erfolgt während der Aufbereitung. Bei Verwendung von **Echtzeit**variablen (z.B. Istwerten) wird die Satzaufbereitung unterbrochen, damit die aktuellen Echtzeitwerte bis zum Vorgängersatz bereitgestellt werden können.
Synchronaktionen werden in aufbereiteter Form mit dem aufbereiteten Satz in den Interpolator transportiert. Die verwendeten Echtzeitvariable werden im **Interpolationstakt** ausgewertet. Die Satzaufbereitung wird nicht unterbrochen.

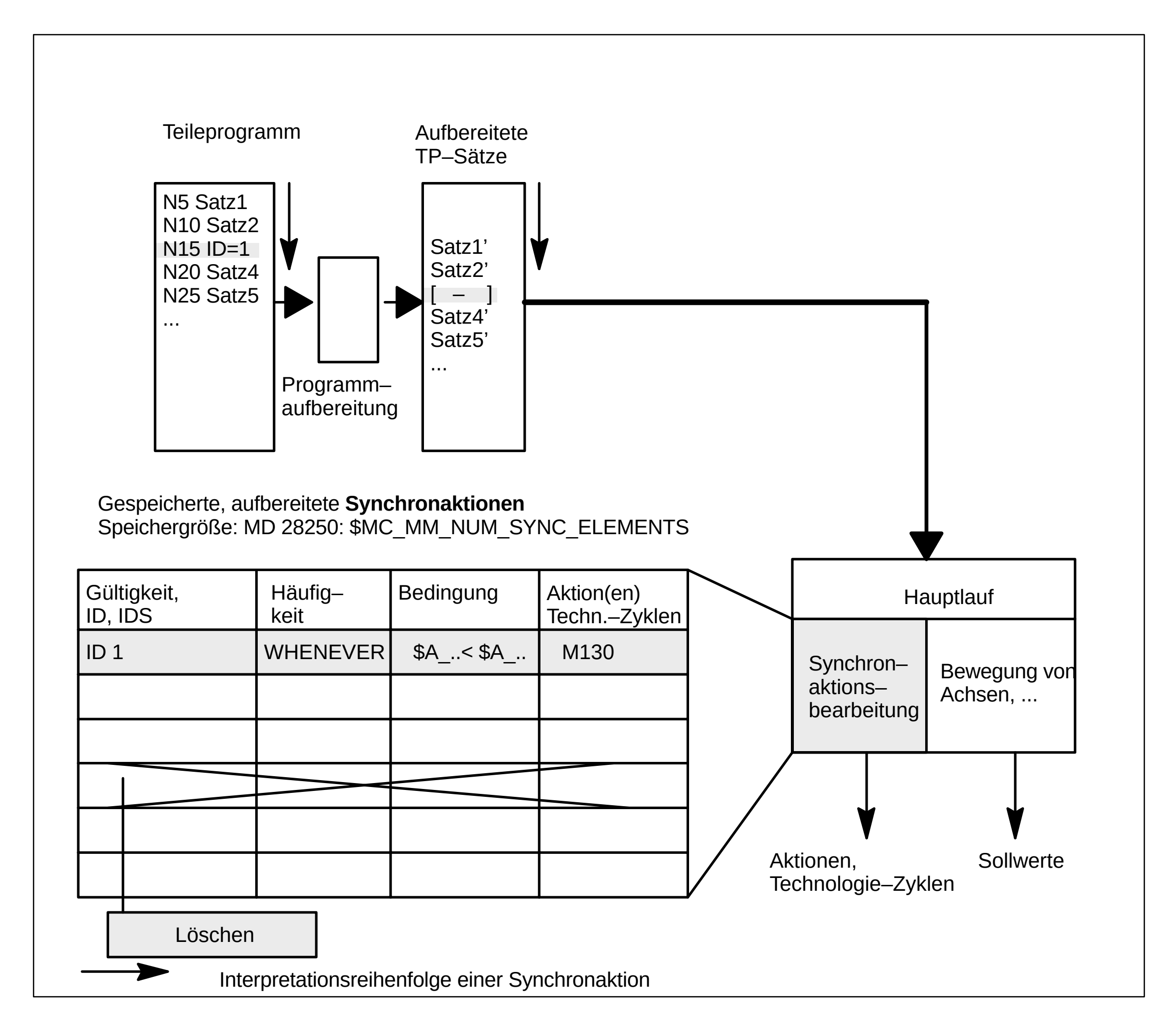



Bild 2-1 Bearbeitung von Synchronaktionen schematisch

**Bearbeitung der Synchronaktionen**

Die Überprüfung, ob Aktionen in Synchronaktionen zu aktivieren sind, erfolgt im Interpolationstakt.
Aktion(en) werden synchron zur Bahnführung ausgeführt, wenn die links der Aktion(en) stehenden Voraussetzungen erfüllt sind.

**Abarbeitungsreihenfolge**

Innerhalb eines Interpolationstaktes werden modal wirksame Synchronaktionsanweisungen in der Reihenfolge ihrer ID–Nummer bearbeitet (Satz mit ID–Nummer 1 vor Satz mit ID–Nummer 2 ...). Nach den modal wirksamen Synchronaktionsanweisungen werden die satzweise wirksamen Synchronaktionsanweisungen in der Reihenfolge ihrer Programmierung bearbeitet.

### 2.1.1 Definition von Bewegungssynchronaktionen

**Definierende Programme**

Bewegungssynchronaktionen können definiert werden:

- im Teileprogramm
- Statische Synchronaktionen in einem asynchronen Unterprogramm ASUP, das durch die PLC aktiviert wird

### 2.1.2 Ausführung der Aktionen

**Ausführungsbedingungen**

Die Aktionen in Bewegungssynchronaktionen werden ausgeführt, wenn:

- die Synchronaktion existiert und nicht abgewählt wurde mit CANCEL(ID). S. 2.5.1
- die Synchronaktion nicht gesperrt ist, kein LOCK(ID). S. 2.5.1
- aufgrund des Häufigkeitsschlüsselwortes eine Auswertung fällig ist
- die Bedingung erfüllt ist

Weitere Details finden Sie in den folgenden Unterkapiteln.

### 2.1.3 Liste möglicher Aktionen

- Ausgabe von M–, S– und H–Hilfsfunktionen an die PLC
- Durch Setzen (Schreiben) von Echtzeitvariablen ist möglich:
  - Überlagerte Bewegung ($AA_OFF), Option.
  - Vorschubbeeinflussung ($AC_OVR, $AA_OVR), Sperren einer programmierten Achsbewegung
  - ...
- Verändern von SW–Nockenpositionen und –zeiten (Settingdaten) und Verändern von anderen Settingdaten
- Veränderung von Koeffizienten und Grenzen aus FCTDEF
- Polynomauswertung SYNFCT
- Online–Werkzeugkorrektur FTOC
- Einlesesperre RDISABLE
- Aufheben Vorlaufstopp STOPREOF
- Restweglöschen DELDTG
- Ermittlung von Kurventabellenwerten
- Axialer Vorschub aus Synchronaktionen
- Achsiale Frames
- Achsen aus Synchronaktionen bewegen/positionieren
- Spindelbewegungen aus Synchronaktionen
- Istwertsetzen aus Synchronaktionen (Preset)
- Kopplungen und Mitschleppen aktivieren, deaktivieren
- Messen aus Synchronaktionen
- Setzen und Löschen von Wartemarken der Kanalsynchronisation
- Alarm setzen/ Fehlerreaktionen
- Fahren auf Festanschlag FXS (FXST, FXSW)
- Fahren mit begrenztem Momemt FOC (FOCON/FOCOF)
- Erweitertes Stillsetzen und Rückziehen (Funktionsbeschreibung M3)
- Lesen und, wenn entsprechend gekennzeichnet, Schreiben von Systemvariablen aus der Liste in 2.3.8.

In Kapitel 2.4 werden diese Aktionen im Detail beschrieben.

## 2.2 Auswertungen und Berechnungen in Echtzeit

**Abgrenzung**

Die Berechnungen, die in Echtzeit durchgeführt werden, sind eine Untermenge der in der NC–Sprache möglichen Berechnungen. Sie ist beschränkt auf die Datentypen REAL, INT, CHAR und BOOL.

ImpliziteTypwandlungen, wie im Teileprogramm, finden nicht statt. Siehe unten Datentyp.

**Anwendungsfeld**

Der Begriff Echtzeit–Ausdruck bezeichnet im Folgenden alle im Interpolations-takt möglichen Berechnungen. Der Echtzeit–Ausdruck wird verwendet in der Bedingung und in der Zuweisung an NC–Adressen und Variablen.

**Echtzeitvariablen**

Alle Echtzeitvariablen werden im Interpolationstakt ausgewertet (gelesen) und können als Bestandteil einer Aktion geschrieben werden.

**Kennzeichen von Echtzeitvariablen**

Echtzeitvariablen sind alle Variablen, die beginnen mit:
**$A**... (Hauptlaufvariable) oder
**$V**... (Servo–Werte).
Zur besonderen Kennzeichnung können diese Variable in Synchronaktionen mit

**$$** programmiert werden.
z.B. $AA_IM[X] oder $$AA_IM[Y]: Istwert für X–Achse oder Y–Achse im Maschinenkoordinatensystem.

---

**Hinweis**

Settingdaten und Maschinendaten müssen mit **$$**S... / **$$**M... programmiert werden, wenn sich ihr Wert während der Bearbeitung ändert.

---

**Datentyp**

Innerhalb eines Ausdrucks in Synchronaktionen können nur Echtzeitvariable **eines Datentyps** miteinander verknüpft werden. Um trotzdem Daten verschiedener Typen zu verarbeiten, können die bereitgestellten Konvertierungsroutinen zur Typangleichung benutzt werden (SW 5.2, siehe Konvertierungsroutinen). Im Gegensatz zum vollen Ausdruck in der NC–Sprache erfolgt die Berechnung im Datentyp der Echtzeitvariablen.

```
... DO $R10 = $AC_PARAM[0]     ; Zulässig REAL, REAL
... DO $R10 = $AC_MARKER[0]    ; nicht erlaubt REAL, INT
```

Die folgenden Beispiele für Auswertungen in Echtzeit waren bereits in SW–Stand 3.2 möglich (Sie benutzen auch nur Echtzeitvariable dieses SW–Standes):

Beispiel 1 für SW 3.2

Auf der linken Seite steht eine in Echtzeit ausgewertete Vergleichsvariable und auf der rechten Seite des Vergleichs keine der zugelassene Echtzeitverarbeitungsvariable, die mit $$ beginnt, sondern ein beliebiger Ausdruck.

WHEN $AA_IM[X] > **$A_INA**[1] DO M120

Während der im Folgesatz programmierten Bewegung wird M120 ausgegeben, wenn der Istwert der X–Achse größer ist als der Wert des analogen Eingangs 1. Dabei wird der Istwert in jedem Interpolationstakt neu ausgewertet, während der Wert des analogen Eingangs zum Interpretations–Zeitpunkt gebildet wird.

Beispiel 2 für SW 3.2

Auf der linken Seite steht eine in Echtzeit ausgewertete Vergleichsvariable und auf der rechten Seite des Vergleichs eine für die Synchronaktion zugelassene Echtzeitvariable, die mit $$ beginnt.

WHEN $AA_IM[X] > **$$A_INA**[1] DO M120

Vergleich des aktuellen Istwerts der X–Achse im IPO–Takt mit dem analogen Eingang 1, da auf der rechten Seite des Vergleichs eine $$–Variable steht.

Beide Variable werden im Interpolationstakt miteinander verglichen.

Beispiel 3 für SW 3.2

Zulässig sind $$–Variable auch auf der linken Seite des Vergleichs.

WHEN **$$AA_IM[X]** > **$$A_INA[1]** DO M120

Identisch zu Beispiel 2. Die linke und rechte Seite wird immer in Echtzeit miteinander verglichen.

**Erweiterungen in SW–Stand 4**

Die in Synchronaktionen zur Verfügung stehenden Echtzeit–Variablen sind in 2.3.8 aufgelistet. Die in Folge–Softwareständen neu hinzugekommenen Systemvariablen sind in der Tabelle entsprechend gekennzeichnet.

- Maschinen– und Settingdaten

Bei Maschinen– und Settingdaten muß für Online–Zugriff $$S... oder $$M... programmiert werden, während der in der Interpretation/Decodierung auszuwertende Zugriff mit einem $–Zeichen eingeleitet wird. Für Zugriff aus Synchronaktionen zugelassene Echtzeitvariablen werden nur mit einem $–Zeichen eingeleitet adressiert.

**Konvertierungsroutinen (SW 5.2)**

In der Synchronaktion gibt es keine implizite Typwandlung von REAL nach INT und umgekehrt. Der Anwender kann jedoch zwei Konvertierungsroutinen **RTOI( ) und ITOR( )** für die Typwandlung explizit aufrufen. Die Funktionen sind

- im Teileprogramm und
- aus der Synchronaktion

aufrufbar.

**ITOR**

REAL ITOR( INT ) – Konvertierung von Integer nach Real

Die Funktion wandelt den übergebenen Integer–Wert in einen Real–Wert um und gibt diesen zurück. Die übergebene Variable wird dabei nicht verändert.

Beispiel:

$AC_MARKER[1] = 561
ID=1 WHEN TRUE DO $AC_PARAM[1] = **ITOR**( $AC_MARKER[1] )

**RTOI**

INT RTOI( REAL ) – Konvertierung von Real nach Integer

Die Funktion RTOI() wandelt den übergebenenen Real–Wert in eine gerundete INT–Zahl um und gibt diese zurück. Liegt der Übergabewert außerhalb des als Integer–Wert eindeutig darstellbaren Bereiches, so wird Alarm 20145 "Bewegungssynchronaktion: Arithmetikfehler" ausgegeben, und die Konvertierung wird nicht durchgeführt. Die übergebene Variable wird dabei nicht verändert.

---

**Hinweis**

Die Funktion RTOI() ist nicht umkehrbar eindeutig, d.h. aus dem Rückgabewert, läßt sich der ursprüngliche Real Wert nicht mehr ermitteln, da die Nachkommastellen bei der Konvertierung verlorengehen!

---

Beispiele RTOI:

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                                        ; Ergebnis: 561
...
$AC_PARAM[1] = –63.867
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                                        ; Ergebnis:–64
...
$AC_MARKER[1]= 10
$AC_PARAM[1] = –6386798797.29
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )
                    ;Ergebnis: Alarm 20145

                    ;$AC_MARKER[1] = 10 (unverändert wegen Alarm)
```

**Implizite Typenwandlung (SW 6.4)**

In Synchronaktionen können ab SW 6.4 Variablen verschiedener Datentypen wie z.B. von REAL nach INT und umgekehrt ohne den Aufruf der Funktion RTOI und ITOR einander zugewiesen werden.

Liegen bei der Konvertierung von REAL nach INTEGER Werte außerhalb des Intervalls [INT_MIN, INT_MAX], dann wird der Alarm 20145 "Bewegungssynchronaktion: Arithmetikfehler" ausgegeben, und die Konvertierung wird nicht durchgeführt.

Beispiele:
bisher

```
$AC_MARKER[1] = 561
ID=1 WHEN TRUE DO $AC_PARAM[1] = ITOR( $AC_MARKER[1] )
```

**ab SW 6.4**

```
$AC_MARKER[1] = 561
ID=1 WHEN TRUE DO $AC_PARAM[1] = $AC_MARKER[1]
```

bisher

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = RTOI( $AC_PARAM[1] )  ; 561
```

**ab SW 6.4**

```
$AC_PARAM[1] = 561.4378
ID=1 WHEN TRUE DO $AC_MARKER[1] = $AC_PARAM[1]          : 561
```

**Grundrechenarten**

Echtzeit–Variablen vom Typ REAL und INT können durch Grundrechenarten:

- Addition
- Subtraktion
- Multiplikation
- Division
- Integer–Division
- Modulo–Division

miteinander verknüpft werden. Es können nur Variable gleichen Typs verknüpft werden.

**Ausdrücke**

Ausdrücke aus Grundrechenarten können geklammert und geschachtelt werden. Siehe Prioritäten von Operatoren auf der Folgeseite.

**Vergleiche**

Möglich sind die Vergleichsoperatoren:

| | |
|---|---|
| **==** | gleich |
| **<>** | ungleich |
| **<** | kleiner |
| **>** | größer |
| **<=** | kleiner oder gleich |
| **>=** | größer oder gleich |

**Boolesche Operatoren**

Möglich sind die booleschen Operatoren:

| | |
|---|---|
| **NOT** | NICHT, |
| **AND** | UND**,** |
| **OR** | ODER**,** |
| **XOR** | exklusives ODER |

**Bitweise Operatoren**

Möglich sind die bitweisen Operatoren:

| | |
|---|---|
| **B_OR** | bitweise ODER |
| **B_AND** | bitweise UND |
| **B_XOR** | bitweises exklusives ODER |
| **B_NOT** | bitweise Negierung |

Operanden sind Variablen und Konstanten vom Typ INT.

**Priorität von Operatoren**

Um bei mehrgliedrigen Ausdrücken das gewünschte Verknüpfungsergebnis zu erhalten, sind die Prioritäten der Operatoren bei Berechnungen und Bedingungen zu berücksichtigen:

| | |
|---|---|
| 1. NOT, B_NOT | Verneinung, bitweise Verneinung |
| 2. *, /, DIV, MOD | Multiplikation, Division |
| 3. +, – | Addition, Substraktion |
| 4. B_AND | bitweise UND |
| 5. B_XOR | bitweise exklusives ODER |
| 6. B_OR | bitweises ODER |
| 7. AND | UND |
| 8. XOR | exklusives ODER |
| 9. OR | ODER |
| 10. | nicht vergeben |
| 11. | Vergleichsoperatoren |
| == | gleich |
| <> | ungleich |
| > | größer |
| < | kleiner |
| >= | größer oder gleich |
| <= | kleiner oder gleich |

und ggf. runde Klammern zu verwenden.
Das Verknüpfungsergebnis von Bedingungen muß vom Typ BOOL sein.

Beispiel einer mehrgliedrigen Bedingung:
WHEN ($AA_IM[X] > WERT) AND ($AA_IM[Y] > WERT1) DO ...

**Funktionen**

Von einer Echtzeit–Variable des Typs REAL kann auch der Funktionswert sin, cos etc. gebildet werden.
Möglich sind die Funkionen:

**SIN**, **COS**, **ABS**, **ASIN**, **ACOS**, **TAN**, **ATAN2, TRUNC**, **ROUND, LN, EXP, ATAN, POT, SQRT, CTAB, CTABINV**

Beispiel:
... DO $AC_PARAM[3]=**COS**($AA_IM[X])

Die Erklärung der Bedeutungen der angegebenen Funktionen stehen in:

**Literatur:** /PG/, Programmieranleitung
/PGA/, Programmieranleitung Arbeitsvorbereitung

**Indizierung**

Der Index einer Echtzeit–Feldvariablen kann wiederum eine Echtzeitvariable sein.

Beispiel:
WHEN ... DO $AC_PARAM[ **$AC_MARKER[1]** ] = 3
Der Index $AC_MARKER[1] wird jeweils im Interpolationstakt aktuell ausgewertet.

Einschränkungen:

- Die Schachtelung der Indizierung mit Echtzeit–Variablen ist nicht erlaubt.
- Von einer Variablen, die nicht in Echtzeit gebildet wird, kann kein Echtzeit–Index gebildet werden. Folgende Programmierung liefert Fehler:
  $AC_PARAM[1]=**$P_EP[$AC_MARKER[0]]**

## 2.3 Spezielle Echtzeitvariablen für Synchronaktionen

Die **vollständige Liste** der in Synchronaktionen ansprechbaren Systemvariablen finden Sie in 2.3.8. Im folgenden werden die Eigenschaften einiger spezieller Echtzeitvariablen vorgestellt:

- Merker–/Zähler–Variablen
  - Kanalspezifische Merker
- Zeiten (Timer)
- Synchronaktionsparameter
- R–Parameter
- Maschinen– und Settingdaten
- FIFO–Variablen (Durchlaufspeicher)

**SW–Stand 4**

Die speziellen Echtzeitvariablen: Zeiten, R–Parameter, Maschinen– und Settingdaten sowie FIFO–Variablen stehen ab SW–Stand 4 zur Verfügung.

### 2.3.1 Merker–/Zähler–Variablen

**Kanalspezifische Merker**

Die Variable **$AC_MARKER[n]** dient als Merker oder Zähler im Datentyp INTEGER.

n: Nummer des Merkers: 0–n
Die Anzahl der Merker pro Kanal wird über das Maschinendatum
MD 28256: NUM_AC_MARKER
eingestellt.

Die Merker sind unter gleichem Namen einmal pro Kanal vorhanden.
Die Merker werden im dynamischen Speicher gehalten. Bei Power On, NC–Reset und Programmende werden die Merker auf 0 gesetzt. Damit sind gleiche Startbedingungen für jeden Programmdurchlauf gegeben.

Die Merker–Variable kann in Synchronaktionen gelesen und beschrieben werden.

**Zusätzlich ab SW 6.3**

Ab SW–Stand 6.3 besteht die Möglichkeit, den Speicherort für $AC_MARKER[n] zwischen DRAM und SRAM mit
MD 28257: MM_BUFFERED_AC_MARKER zu wählen.

0: dynamischer Speicher DRAM, (Default)
1: statischer Speicher SRAM

Im MD 28256: NUM_AC_MARKER kann als Höchstwert 20000 angegeben werden. Ein Element benötigt 4 Bytes Speicherplatz. Es muß darauf geachtet werden, daß der erforderliche Speicherplatz in der gewählten Speicherart verfügbar ist.

Im SRAM gespeicherte Merker können in die Datensicherung einbezogen werden. Siehe 2.3.7

## 2.3.2 Zeiten (Timer)

Die Systemvariable **$AC_TIMER[n]** ermöglicht das Starten von Aktionen nach definierten Wartezeiten.

n: Nummer der Timer–Variable
Einheit: Sekunde
Datentyp: REAL

Die Anzahl der verfügbaren Timer–Variablen wird per Maschinendatum

MD 28258: MM_NUM_AC_TIMER

festgelegt.

**Timer setzen**

Das Hochzählen einer Timer–Variable wird gestartet durch Wertzuweisung:

$AC_TIMER[n]=value
n: Nummer der Zeitvariable
value: Startwert (i.d.R. 0)

**Timer anhalten**

Das Hochzählen einer Timer–Variable wird gestoppt durch Zuweisung eines negativen Wertes:

$AC_TIMER[n]=–1

**Timer lesen**

Der aktuelle Zeitwert kann bei laufender oder gestoppter Timer–Variablen gelesen werden. Nach dem Stoppen der Timer–Variablen durch Zuweisung von –1 bleibt der zuletzt aktuelle Zeitwert stehen und kann weiterhin gelesen werden.

**Beispiel**

Ausgabe eines Istwertes über Analogausgang 500ms nach Erkennen eines digitalen Eingangs:

WHEN $A_IN[1]==1 DO **$AC_TIMER[1]=0** ;Timer rücksetzen und starten
WHEN **$AC_TIMER[1]>=0.5** DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=–1

## 2.3.3 Synchronaktionsparameter

Die Variablen **$AC_PARAM[n]** dienen als Zwischenspeicher in Synchronaktionen.

Datentyp: REAL
n: Nummer des Parameters 0 – n

Die Anzahl der verfügbaren AC–Parameter–Variablen pro Kanal wird über das Maschinendatum

MD 28254: MM_NUM_AC_PARAM

festgelegt.

Die Parameter sind unter gleichem Namen einmal pro Kanal vorhanden. Die Parameter $AC_PARAM werden im dynamischen Speicher gehalten. Bei Power On, NC–Reset und Programmende werden die Parameter auf 0 gesetzt. Damit sind gleiche Startbedingungen für jeden Teileprogrammdurchlauf gegeben. Die Variablen $AC_PARAM können in Synchronaktionen gelesen und beschrieben werden.

**Zusätzlich ab SW 6.3**

Ab SW–Stand 6.3 besteht die Möglichkeit, den Speicherort für $AC_PARAM[n] zwischen DRAM und SRAM mit MD 28255: MM_BUFFERED_AC_PARAM zu wählen.

0: dynamischer Speicher DRAM, (Default)
1: statischer Speicher SRAM

Im MD 28255: NUM_AC_PARAM kann als Höchstwert 20000 angegeben werden. Ein Element benötigt 8 Bytes Speicherplatz. Es muß darauf geachtet werden, daß der erforderliche Speicherplatz in der gewählten Speicherart verfügbar ist.

Im SRAM gespeicherte Synchronaktionsparameter können in die Datensicherung einbezogen werden. Siehe 2.3.7

## 2.3.4 R–Parameter

**Definition**

R–Parameter sind Variablen vom Typ REAL, die im batteriegepufferten Speicher gehalten werden.

Deshalb behalten sie über Programmende, RESET, Power ON hinweg ihre Werte.

**Verwendung in Synchronaktionen**

Mit dem Einleitungszeichen **$** vor dem R–Parameter können R–Parameter auch in Synchronaktionen verwendet werden.

Beispiel:
WHEN $AC_MEA== 1 DO **$R10**= $AA_MM[Y]
; wenn gültige Messung vorliegt, Meßwert in R–Parameter übernehmen

---

**Hinweis**

Es wird empfohlen, eine bestimmte R–Variable entweder normal im Teileprogramm oder in Synchronaktionen zu verwenden. Soll eine in Synchronaktionen verwendete R–Variable anschließend wieder "normal" im Teileprogramm verwendet werden, so muß durch STOPRE für die Synchronisation gesorgt werden. Beispiel:

```
WHEN $A_IN[1] == 1 DO $R10 = $AA_IM[Y]
G1 X100 F150
STOPRE
IF R10 > 50 ....     ; Auswertung des R–Parameters
```

---

## 2.3.5 Maschinen– und Settingdaten

Ab SW–Stand 4 ist das Lesen und Schreiben von Maschinen– und Settingdaten aus Synchronaktionen möglich. Der Zugriff muß unterschieden werden nach:

- MD, SD, die während der Bearbeitung unverändert bleiben
- MD, SD, die sich während der Bearbeitung verändern

**Unveränderliche MD, SD lesen**

Unveränderliche Maschinendaten und Settingdaten werden aus der Synchronaktion adressiert wie in normalen Teileprogramm–Befehlen. Sie werden mit einem **$**–Zeichen eingeleitet.
Beispiel:
ID=2 WHENEVER $AA_IM[z]< **$**SA_OSCILL_REVERSE_POS2[Z]–6 DO $AA_OVR[X]=0
; Hier wird der während der Bearbeitung als unveränderlich angenommene
; Umkehrbereich 2 für Pendeln angesprochen

Ein vollständiges Beispiel für Pendeln mit Zustellung im Umkehrbereich finden Sie in Kap. 6.2 und:

**Literatur:** /FB/, P5, Pendeln

**Veränderliche MD, SD lesen**

Während der Bearbeitung sich ändernde Maschinendaten und Settingdaten werden aus der Synchronaktion mit **$$**–Zeichen eingeleitet adressiert.

Beispiel:
ID=1 WHENEVER $AA_IM[z]< **$$**SA_OSCILL_REVERSE_POS2[Z]–6 DO $AA_OVR[X]=0
In diesem Zusammenhang wird davon ausgegangen, daß die Umkehrposition durch Bedienung jederzeit verändert werden könnte.

**MD, SD schreiben**

Voraussetzung:
Das aktuell eingestellte Zugriffsrecht muß den Schreibzugriff zulassen. Es ist nur sinnvoll, MD und SD aus der Synchronaktion zu verändern, wenn die Änderung sofort wirksam wird. Die Wirksamkeit nach Änderung wird für alle MD und Settingdaten angegeben in:

**Literatur:** /LIS/, Listen

Adressierung:
Zu änderden Maschinendaten und Settingdaten sind eingeleitet mit **$$** zu adressieren.

Beispiel:

ID=1 WHEN $AA_IW[X]>10 DO **$$**SN_SW_CAM_PLUS_POS_TAB_1[0]= 20
**$$**SN_SW_CAM_MINUS_POS_TAB_1[0]= 30
; Veränderung der Schaltpositionen von SW–Nocken

## 2.3.6 FIFO–Variablen (Durchlaufspeicher)

**Anwendung**

Zur Abspeicherung zusammengehöriger Datenfolgen stehen bis zu 10 FIFO–Variablen zur Verfügung: **$AC_FIFO1[n]** bis **$AC_FIFO10[n]** .

**Struktur**

Die Speicherstruktur einer FIFO–Variablen zeigt Bild 2-3 .

**Anzahl**

Die Anzahl der verfügbaren AC–FIFO–Variablen wird per Maschinendatum
MD 28260: NUM_AC_FIFO
festgelegt.

**Größe**

Die Anzahl der in eine FIFO–Variable ablegbaren Werte wird durch das Maschinendatum
MD 28264: LEN_AC_FIFO
definiert. Alle FIFO–Variablen haben gleiche Länge.

**Datentyp**

Werte in der FIFO–Variablen haben den Datentyp REAL.

**Bedeutung Index**

Index n:

Die Indizes 0 bis 5 haben Sonderbedeutungen:

n= 0: Beim Schreiben mit Index 0 wird ein neuer Wert in den FIFO abgelegt
Beim Lesen mit Index 0 wird das älteste Element gelesen und aus dem FIFO entfernt

n=1: Zugriff auf das älteste gespeicherte Element
n=2: Zugriff auf das jüngste gespeicherte Element
n=3: Summe aller FIFO–Elemente
Das MD 28266: MODE_AC_FIFO bestimmt den Modus der Summenbildung:
Bit 0 = 1 Summe bei jedem Einschreiben aktualisieren
Bit 0 = 0 Keine Summenbildung

n=4: Anzahl der im FIFO verfügbaren Elemente.
Auf jedes Element des FIFO kann lesend und schreibend zugegriffen werden.
Das Rücksetzen der FIFO–Variablen erfolgt durch Rücksetzen der Element–Anzahl z.B. für die erste FIFO–Variable: **$AC_FIFO1[4]=0**
n=5 aktueller Schreibindex relativ zum FIFO–Anfang

n= 6 bis 6+nmax:
Zugriff auf n–tes FIFO–Element:

---

**Hinweis**

Der FIFO–Zugriff ist eine spezielle Form des R–Parameter–Zugriffs: (s. unten)
Die FIFO–Werte werden im R–Parameterbereich hinterlegt.
Die FIFO–Werte liegen im statischen Speicher. Sie bleiben über Programmende / Reset und Power On hinweg erhalten. Bei der Archivierung von R–Parametern werden die FIFO–Werte mit gesichert.

---

Das Maschinendatum
MD 28262: START_AC_FIFO
legt die Nummer des R–Parameters fest, ab der FIFO–Variablen im R–Parameter–Bereich liegen.

Die aktuelle Anzahl für R–Parameter eines Kanals wird durch das Maschinendatum
MD 28050: MM_NUM_R_PARAM
definiert.

Die zwei folgenden Bilder zeigen Teilelängen von Teilen auf einem Band, die in einer FIFO–Variablen abgelegt wurden schematisch.

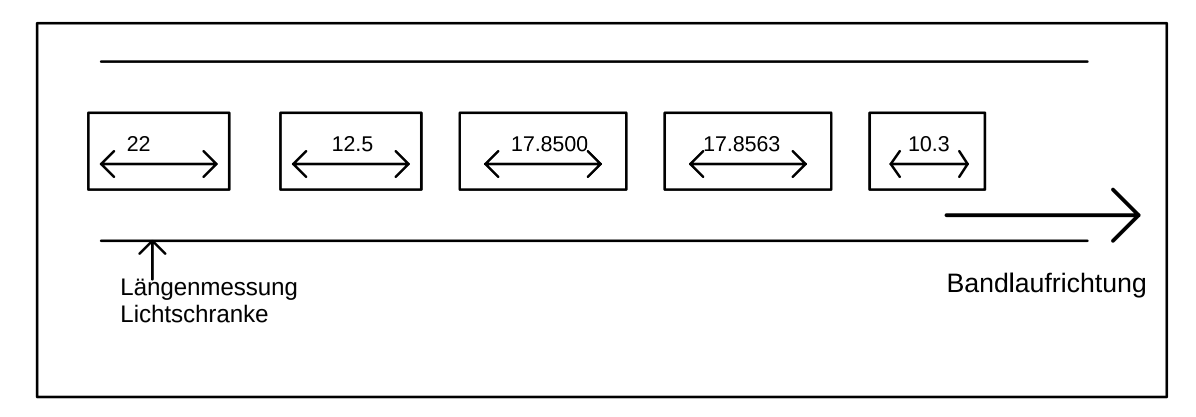



Bild 2-2 Produktlängen einer Teilefolge auf Band

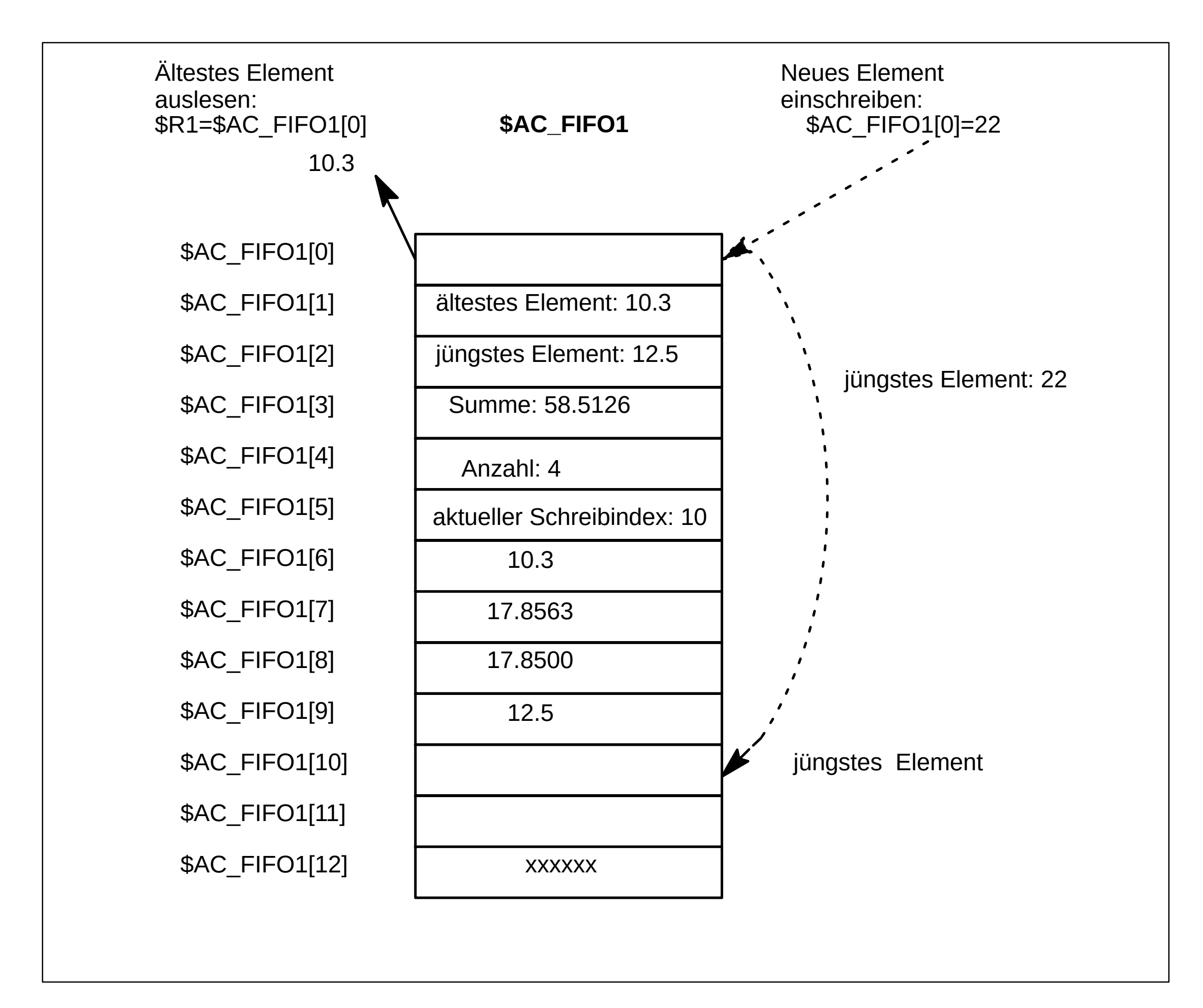



Bild 2-3 Beispiel FIFO–Variable

### 2.3.7 SRAM gespeicherte Systemvariablen (ab SW 6.3)

**RESET–Verhalten**

Im SRAM gespeicherte Systemvariablen $AC_MARKER und $AC_PARAM behalten über RESET und Power On hinweg ihre aktuellen Werte.

---

**Hinweis**

In Teileprogrammen und Synchronaktionen, die mit SRAM–gespeicherten Systemvariablen arbeiten, muß beachtet werden, daß nach RESET keine Initialisierung der Variablen mit 0 stattfindet. Das erfordert ggf. Anpassungen, wenn zuvor mit DRAM gespeicherten Systemvariablen gearbeitet worden ist.

---

**Datensicherung**

Im SRAM gespeicherte Systemvariablen $AC_MARKER und $AC_PARAM können in die Datensicherung einbezogen werden. Pro Kanal existieren die Sicherungsbausteine:

| | |
|---|---|
| _N_CHi_ACM | für $AC_MARKER Werte und |
| _N_CHi_ACP | für $AC_PARAM Werte. |

i steht für die jeweilige Kanalnummer.

**Reihenfolge**

Die gesicherten Bausteine werden im File der Gesamtsicherung _N_INITIAL_INI nach R–Parametern eingetragen.

**Literatur:** /IAD/, Inbetriebnahmeanleitung

### 2.3.8 Liste der für Synchronaktionen bedeutsamen Systemvariablen

**Übersicht**

In der folgenden Tabelle finden Sie, sortiert nach Kategorien, diejenigen Systemvariablen, die aus Synchronaktionen heraus lesend oder schreibend angesprochen werden können. Die Zugriffsmöglichkeiten sind angegeben.

**Legende:**

| | |
|---|---|
| r | Lesen |
| w | Schreiben |
| R | Lesen mit implizitem Vorlaufstop |
| W | Schreiben mit implizitem Vorlaufstop |
| TP | Teileprogramm |
| SA | Synchronaktion |
| SW | SW–Stand s. Hinweis |

**Hinweis**

Unter **Typ** steht dann der SW–Stand (z.B. /4) der Einführung der Systemvariablen, wenn diese nicht schon seit SW 2 besteht.

Unter **Zugriff SA** steht dann der SW–Stand der Einführung der **Zugriffsmöglichkeit** aus Synchronaktionen auf die Systemvariable, wenn diese nicht schon seit Einführung der Systemvariablen besteht.
Der Namensteil ”ACT” in Systemvariablen für Synchronaktionen (z.B. $AA_V**ACT**M) kennzeichnet **Soll**werte, die im Interpolator berechnet werden und als Eingangsgrößen für die Achsregelung verwendet werden.
Der Namensbeginn ”$VA_...” kennzeichnet echte Istwerte, die über die Auswertung der Encoder–Information tatsächliche Werte einer Maschinenachse wiedergeben.

**Anwendervariablen**

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_MARKER[n]** | INT | Merkervariable für Bewegungssynchronaktionen | Zähler | R/W | r/w |
| **$AC_PARAM[n]** | DOUBLE | Rechenvariable f. Bewegungssynchronaktionen | Zähler | R/W | r/w |
| **$AC_FIFOi[n]** | DOUBLE /4 | i: 1–10,<br>Nr. der FIFO–Variablen, (Durchlaufspeicher)<br>n: Parameter–Nummer , 0 – max. Fifo–Element.<br>Bedeutungen n:<br>n=0: Beim Schreiben mit Index 0 wird ein neuer Wert in den FIFO abgelegt. Beim Lesen mit Index 0 wird das älteste Element gelesen und aus dem FIFO entfernt.<br>n=1: Lesezugriff auf ältestes Element<br>n=2: Lesezugriff auf jüngstes Element<br>n=3: Summe aller im FIFO befindlichen Elemente, wenn im Md $MC_MM_MODE_FIFO Bit0 gesetzt ist.<br>n=4: Lesezugriff auf die aktuelle Anzahl der FIFO–Elemente<br>n= 5 – m: Lese–Zugriff auf einzelne FIFO–Elemente. 5 ist das älteste Element, 6 das zweitälteste etc. | Nr. Parameter | R/w | r/w |

### R–Parameter

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$R[n], Rn** | DOUBLE | die max. Anzahl der R–Parameter wird per Maschinendatum festgelegt.<br>Aus Synchronaktionen werden R–Parameter angesprochen mit $R oder $R[i]. Sonst ...Rn oder R[n] | Zähler | r/w | r/w /4 |

### Systemdaten

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AN_SETUP_TIME** | DOUBLE | IF $AN_SETUP_TIME > 60000 GOTOF MARK01<br>Zeit seit dem letztenSteuerungshochlauf mit Default–Daten. | Minuten | R | r /5 |
| **$AN_POWERON_TIME** | DOUBLE | IF $AN_POWERON_TIME == 480 GOTOF MARK02<br>Zeit seit dem letztenSteuerungs–Normalhochlauf. | Minuten | R | r /5 |
| **$AN_NCK_VERSION** | DOUBLE | NCK–Version:<br>Ausgewertet wird nur der Vorkommaanteil der Gleitzahl, der Nachkommaanteil kann Kennungen für entwicklungsinterne Zwischenstände enthalten. Der Vorkommaanteil beeinhaltet die offizielle Softwarestandskennung des NCK: Beispielsweise ist für den NCK–Stand 20.00.00 ist der Wert der Variablen 200000,0. | NCK–Version | R | r /6 |

### Werkzeugdaten

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_TOOLMN[t]** | INT | Magazin – Nummer von WZ t | T–Nummer | R | r /4 |
| **$A_TOOLMLN[t]** | INT | Magazin – Nummer von WZ t | T–Nummer | R | r /4 |
| **$A_MONIFACT** | REAL | Faktor für Standzeitüberwachung | | R/W | r /4 |
| **$AC_MONMIN** | REAL | Verhältnis von Werkzeugüberwachungsistwert zu Sollwert. Schwelle für die Werkzeugsuchstrategie ”nur Werkzeuge mit Istwert größer als Schwelle einwechseln” | | R/W | r /5 |
| **$A_DNO[i]** | INT | Lesen einer von PLC vorgegebenen D–Nummer über VDI–Nahtstelle | Index | R | r /4 |

### G–Gruppen

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_GG** | INT | $A_GG[n]<br>Aktuelle G–Funktion einer Gruppe aus Synchronaktion lesen<br>n: Nummer der G–Gruppe | wie PLC–Nahtstelle |  | r /5 |
| **$P_ACTID[n]** | BOOL | Modale Synchronaktion mit ID n aktiv, wenn TRUE<br>n: 1 – 16 | Synchronaktion mit ID | R | r /2 |
| **$P_GG[n]** | INT | aktuelle G–Funktion einer Gruppe aus TP lesen<br>n: Nummer der G–Gruppe | wie PLC Nahtstelle | R | /2 |

### Kanalzustände

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AC_STAT** | INT<br>/4 | Kanalzustand<br>–1: ungültig<br>0: Kanal im Reset<br>1: Kanal unterbrochen<br>2: Kanal aktiv |  | R | r /4 |
| **$AC_PROG** | INT<br>/4 | Programmzustand<br>–1: ungültig<br>0: Programm im Resetzustand<br>1: Programm gestoppt<br>2: Programm aktiv<br>3: Programm wartend<br>4: Programm unterbrochen |  | R | r /4 |
| **$AC_SYNA_MEM** | INT<br>/4 | Freispeicher für Bewegungssynchronaktionen zeigt an, wieviele Elemente des mit $MC_MM_NUM_SYNC_ELEMENTS belegten-Speichers noch frei sind. |  | R | r /4 |
| **$AC_IPO_BUF** | INT<br>/4 | Füllstand des Interpolations–Puffers |  | R | r /4 |
| **$AC_IW_STAT** | INT<br>/5 | Stellungsinformation der Gelenke (transformationsspezifisch) für kart PTP Fahren |  | R | r /5 |
| **$AC_IW_TU** | INT<br>/5 | Stellungsinformation der Achsen ( MCS ) für kart PTP Fahren |  | R | r /5 |
| **$A_PROBE[n]** | INT<br>/4 | $A_PROBE[1]: Zustand erster Meßtaster<br>$A_PROBE[2]: Zustand zweiter Meßtaster<br>0: nicht ausgelenkt<br>1: ausgelenkt | Nr. Taster | R | r |
| **$AC_MEA[n]** | INT | Meßtaster hat geschaltet, wenn TRUE (1)<br>1 – MAXNUM_PROBE | Nr. Taster | R | r /4 |
| **$AC_TRAFO** | INT | Codenummer der aktiven Transformation entsprechend Maschinendatum $MC_TRAFO_TYPE n | – | R | r /4 |
| **$AC_LIFTFAST** | INT | Schnellabheben:<br>0: Es war kein Rückhub aktiv<br>1: Es war Rückhub aktiv | – | R/W | r/w |

## ASUPs

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_ASUP** | INT<br>/4 | Siehe auch: **Literatur:** /FB/, K1,<br>(BAG, Kanal, Programmbetrieb)<br>Anmerkungen:<br>Grund der Aktivierung eines Asups.<br>Akt. wg.: Aktivierung wegen ...<br>Akt. durch: Aktivierung durch ...<br>Forts.: Fortsetzmöglichkeit(n) ...<br>**BIT 0:**<br>Akt. wg.: Anwender–Interrupt ”ASUP mit Blsync” (Satzsynchronisation),<br>Akt. durch: Vdi–Signal, Dig.–Anal. Schnittstelle,<br>Forts. : freiwählbar Reorg oder Ret<br>**BIT 1:**<br>Akt. wg.: Anwender–Interrupt ”ASUP”. (Für die Programm–Fortsetzung mit Repos wird die Position nach dem Satz, in dem gestopped wurde, abgespeichert.)<br>Akt. durch: Vdi–Signal, Dig.–Anal. Schnittstelle<br>Forts.: freiwählbar Reorg oder Ret<br>**BIT 2:**<br>Akt. wg.: Anwender–Interrupt ”ASUP aus Kanalzustand Ready”,<br>Akt. durch:Vdi–Signal, Dig.–Anal. Schnittstelle<br>Forts.: freiwählbar Reorg oder Ret<br>**BIT 3:**<br>Akt. wg: Anwender–Interrupt ”ASUP in einer Handbetriebsart und Kanalzustand nicht READY ”<br>Akt. durch: Vdi–Signal, Dig.–Anal. Schnittstelle<br>Forts. durch: freiwählbar Reorg oder Ret<br>**BIT 4:**<br>Akt. wg: Anwender–Interrupt ”ASUP”. (Für die Programm–Fortsetzung mit Repos wird die aktuelle Position beim Auftreten des Interrupts abgespeichert).<br>Akt. durch: Vdi–Signal, Dig.–Anal. Schnittstelle<br>Forts. durch: freiwählbar Reorg oder Ret<br>**BIT 5:** Akt. wg.: Abbrechen der Unterprogrammwiederholung<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 6:**<br>Akt. wg.: Aktivierung Decodier–Einzelsatz<br>Akt. durch: Vdi–Signal (+BTSS)<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 7:** Akt. wg.: Aktivierung Restweglöschen<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup Ret<br>**BIT 8:**<br>Akt. wg: Aktivierung Achssynchronisation<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 9:**<br>Akt. wg.: Betriebsartenwechsel<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS oder RET (siehe MD.) | | R | r /4 |

| Name | Typ /SW | Beschreibung/Werte | Index | Zu- griff TP | Zu- griff SA /SW |
|---|---|---|---|---|---|
| **$AC_ASUP** ff. | INT /4 | **BIT 10:** Akt. wg.: Programmfortsetzung unter TeachIn bzw. nach TeachIn–Deaktivierung<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup Ret<br>**BIT 11:**<br>Akt. wg.: Overstore Anwahl<br>Akt. durch: Pi–Dienst<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 12:**<br>Akt. wg.: Alarm mit Reaktion Korrektursatz mit Repos<br>Akt: durch: Intern<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 13:** Akt. wg.: Rückzugbewegung bei G33 und Stop<br>Akt. durch: Intern<br>Forts. durch: Einsatz des System Asup Ret<br>**BIT 14:**<br>Akt. wg.: Aktivierung von Probelauf Vorschub<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 15:**<br>Akt. wg.: Deaktivierung von Probelauf Vorschub<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 16:**<br>Akt. wg.: Aktivierung von Satzunterdrückung<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 17:**<br>Akt. wg.: Deaktivierung von Satzunterdrückung<br>Akt. durch: Vdi–Signal<br>Forts. durch: Einsatz des System Asup REPOS<br>**BIT 18:**<br>Akt. wg: Maschinendaten–wirksam setzen<br>Akt. durch: Pi–Dienst<br>Forts. durch: Einsatz des System Asup REPOS<br>**Bit 19**:<br>Akt. wg: Werkzeugkorrektur wirksam setzen<br>Akt. durch: Pi–Dienst ”_N_SETUDT”<br>Forts. durch: Einsatz des System Asup REPOS<br>**Bit 20**:<br>Akt. wg: SystemAsup nachdem Suchlauftyp SERUPRO das Suchziel erreicht hat.<br>Akt. durch: Pi–Di. ”_N_FINDBL” Parameter == 5<br>Forts. durch: Einsatz des System Asup REPOS | | R | r /4 |

## Komandos an / von Kanal (Diagnose)

| Name | Typ /SW | Beschreibung/Werte | Index | Zu- griff TP | Zu- griff SA /SW |
|---|---|---|---|---|---|
| **$A_PROTO** | BOOL /4 | Protokollierfunktion für den ersten User aktivie- ren / deaktivieren<br>0: ausschalten<br>1: einschalten | | R/W | r/w |
| **$A_PROTOC** | BOOL /4 | Protokollierfunktion für einen User aktivieren / deaktivieren<br>0: ausschalten<br>1: einschalten | | R/W | r/w |

### Ein–/ Ausgänge

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_IN[n]** | BOOL | digitaler Eingang der NC | Nr. Eingang | R | r |
| **$A_OUT[n]** | BOOL | digitaler Ausgang der NC | Nr. Ausgang | R/w | r/w |
| **$A_INA[n]** | DOUBLE | analoger Eingang der NC | Nr. Eingang | R | r |
| **$A_OUTA[n]** | DOUBLE | analoger Ausgang der NC<br>Beim Schreiben wird der Wert erst mit dem nächsten Ipo–Takt wirksam und kann dann wieder zurück gelesen werden. | Nr. Ausgang | R/w | r/w |
| **$A_INCO[n]** | BOOL | Comparator–Eingang | Nr. Eingang | R | r |

### Schreiben und Lesen von PLC–Variablen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_DBB[n]** | INT /4 | Datenbyte (8Bit) von/an PLC lesen/schreiben | Offset im E/A–Bereich | R/w | r/w |
| **$A_DBW[n]** | INT /4 | Datenwort (16Bit) von/an PLC lesen/schreiben | Offset im E/A–Bereich | R/w | r/w |
| **$A_DBD[n]** | INT /4 | Datendoppelwort (32Bit) von/an PLC lesen/ schreiben | Offset im E/A–Bereich | R/w | r/w |
| **$A_DBR[n]** | DOUBLE /4 | Realdaten (32Bit) von/an PLC lesen/schreiben | Offset im E/A–Bereich | R/w | r/w |

### Linkvariablen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_DLB[n]** | INT /5 | Datenbyte (8Bit) von/an NCU–Link lesen/schreiben | Positionsoffset im Link–Speicher | R/w | r/w |
| **$A_DLW[n]** | INT /5 | Datenwort (16Bit) von/an NCU–Link lesen/ schreiben | Positionsoffset im Link–Speicher | R/w | r/w |
| **$A_DLD[n]** | INT /5 | Datendoppelwort (32Bit) von/an NCU–Link lesen/schreiben | Positionsoffset im Link–Speicher | R/w | r/w |

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_DLR[n]** | DOUBLE /5 | Real Daten (32Bit) von/an NCU–Link lesen/ schreiben | Positionsoffset im Link–Speicher | R/w | r/w |
| **$A_LINK_TRANS _RATE** | INT /5 | Anzahl Bytes, die noch im aktuellen Ipo–Takt über die NCU–Link–Kommunikation übertragen werden können. | – | r/ | r/ |

## Direkt PLC_IO

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_PBB_IN[n]** | INT /5 | Datenbyte (8Bit) direkt von PLC–IO lesen | Positionsoffset im PLC–Eingangs–Bereich | R/ | r/ |
| **$A_PBW_IN[n]** | INT /5 | Datenwort (16Bit) direkt von PLC–IO lesen | Positionsoffset im PLC–Eingangs–Bereich | R/ | r/ |
| **$A_PBD_IN[n]** | INT /5 | Datendoppelwort (32Bit) direkt von PLC–IO lesen | Positionsoffset im PLC–Eingangs–Bereich | R/ | r/ |
| **$A_PBR_IN[n]** | DOUBLE /5 | Real Daten (32Bit) direkt auf PLC–IO lesen | Positionsoffset im PLC–Eingangs–Bereich | R/w | r/w |
| **$A_PBB_OUT[n]** | INT /5 | Datenbyte (8Bit) direkt auf PLC–IO schreiben | Positionsoffset im PLC–Ausgangs–Bereich | R/w | r/w |
| **$A_PBW_OUT[n]** | INT /5 | Datenwort (16Bit) direkt auf PLC–IO schreiben | Positionsoffset im PLC–Ausgangs–Bereich | R/w | r/w |

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_PBD_OUT[n]** | INT /5 | Datendoppelwort (32Bit) direkt auf PLC–IO schreiben | Positionsoffset im PLC–Ausgangs–Bereich | R/w | r/w |
| **$A_PBR_OUT[n]** | DOUBLE /5 | Real Daten (32Bit) direkt auf PLC–IO schreiben | Positionsoffset im PLC–Ausgangs–Bereich | R/w | r/w |

## Werkzeugverwaltung

**Literatur:** /FBW/, Werkzeugverwaltung

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AC_TC_FCT** | INT /5 | Kommandonummer. Diese spezifiziert, welcher Vorgang gewünscht wird. | – | R/ | r/ |
| **$AC_TC_STATUS** | INT /5 | Status, in dem sich das Kommando – zu lesen über $AC_TC_FCT – befindet. | – | R/ | r/ |
| **$AC_TC_THNO** | INT /5 | Nummer des Werkzeughalters (im spez. die Spindelnr.) auf den das neue Werkzeug eingewechselt werden soll. | – | R/ | r/ |
| **$AC_TC_TNO** | INT /5 | NCK–interne T–Nummer des neuen (einzuwechselnden) Werkzeugs.<br>0: es gibt kein neues Werkzeug. | – | R/ | r/ |
| **$AC_TC_MFN** | INT /5 | Quell–Magazinnummer des neuen Werkzeugs.<br>0: es gibt kein neues Werkzeug. | – | R/ | r/ |
| **$AC_TC_LFN** | INT /5 | Quell–Platznummer des neuen Werkzeugs.<br>0: es gibt kein neues Werkzeug. | – | R/ | r/ |
| **$AC_TC_MTN** | INT /5 | Ziel–Magazinnummer des neuen Werkzeugs.<br>0: es gibt kein neues Werkzeug. | – | R/ | r/ |
| **$AC_TC_LTN** | INT /5 | Ziel–Platznummer des neuen Werkzeugs.<br>0: es gibt kein neues Werkzeug. | – | R/ | r/ |
| **$AC_TC_MFO** | INT /5 | Quell–Magazinnummer des alten (auszuwechselnden) Werkzeugs.<br>0: es gibt kein altes Werkzeug. | – | R/ | r/ |
| **$AC_TC_LFO** | INT /5 | Quell–Platznummer des alten (auszuwechselnden) Werkzeugs.<br>0: es gibt kein altes Werkzeug. | – | R/ | r/ |
| **$AC_TC_MTO** | INT /5 | Ziel–Magazinnummer des alten (auszuwechselnden) Werkzeugs.<br>0: es gibt kein altes Werkzeug. | – | R/ | r/ |
| **$AC_TC_LTO** | INT /5 | Ziel–Platznummer des alten (auszuwechselnden) Werkzeugs.<br>0: es gibt kein altes Werkzeug. | – | R/ | r/ |

## Zeiten

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_YEAR** | INT | Systemzeit Jahr | | R | r /4 |
| **$A_MONTH** | INT | Systemzeit Monat | | R | r /4 |
| **$A_DAY** | INT | Systemzeit Tag | | R | r /4 |
| **$A_HOUR** | INT | Systemzeit Stunde | | R | r /4 |
| **$A_MINUTE** | INT | Systemzeit Minute | | R | r /4 |
| **$A_SECOND** | INT | Systemzeit Sekunde | | R | r /4 |
| **$A_MSECOND** | INT | Systemzeit Millisekunde | | R | r /4 |

| | | | | | |
|---|---|---|---|---|---|
| **$AC_TIME** | DOUBLE | Zeit vom Satzanfang in Sekunden | | R | r /4 |
| **$AC_TIMEC** | DOUBLE | Zeit vom Satzanfang in Interpolationstakten | | R | r /4 |
| **$AC_TIMER[n]** | DOUBLE /4 | Timer – Einheit in Sekunden. Die Zeit wird intern gerastert in Vielfachen des Interpolationstaktes gezählt. Das Hochzählen der Zeitvariable wird gestartet durch die Wertzuweisung $AC_TIMER[n]=<startwert>. Das Hochzählen einer Zeitvariable wird gestoppt durch Zuweisung eines negativen Wertes: $AC_TIMER[n]=–1<br>Der aktuelle Zeitwert kann bei laufender oder gestoppter Zeitvariable gelesen werden. Nach dem Stoppen der Zeitvariable durch Zuweisung von –1 bleibt der zuletzt aktuelle Zeitwert stehen und kann weiterhin gelesen werden.<br>Die Dimesion wird über das MD 28258: MM_NUM_AC_TIMER festgelegt. | Zähler | R/W | r/w |

## Bahnbewegung

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_PATHN** | DOUBLE | Normierter Bahnparameter, Wert zwischen 0=Satzanfang und 1=Satzende | | R | r |
| **$AC_DTBW** | DOUBLE | Geometrischer Abstand vom Satzanfang im Werkstückkoordinatensystem. Für die Abstandsberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | | R | r |
| **$AC_DTBB** | DOUBLE | Geometrischer Abstand vom Satzanfang im Basiskoordinatensystem. Für die Abstandsberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | | R | r |
| **$AC_DTEW** | DOUBLE | Geometrischer Abstand vom Satzende im Werkstückkoordinatensystem. Für die Abstandsberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | | R | r |
| **$AC_DTEB** | DOUBLE /3 | Geometrischer Abstand vom Satzende im Basiskoordinatensystem. Für die Abstandsberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | | R | r |
| **$AC_PLTBB** | DOUBLE /3 | Bahnweg vom Satzanfang im Basiskoordinatensystem<br>Die Variable ist nur aus Sychronaktionen zugreifbar | | R | r |
| **$AC_PLTEB** | DOUBLE | Bahnweg zum Satzende im Basiskoordinatensystem<br>Die Variable ist nur aus Sychronaktionen zugreifbar | | R | r |
| **$AC_DELT** | DOUBLE | Restweg Bahn im Werkstückkoordinatensystem nach Restweglöschen bei Bewegungssynchronaktionen | | R | r |
| **$P_APDV** | BOOL | liefert TRUE, wenn die mit $P_APR[X] bzw. $P_AEP[X] lesbaren Positionswerte (Aufstartpunkt bzw. Konturpunkt beim Weichen An– und Abfahren) gültig sind. | | R | r /4 |

## Geschwindigkeiten kanalspezifisch

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$P_F** | DOUBLE | Zuletzt programmierter Bahnvorschub | | R | r |
| **$AC_OVR** | DOUBLE | Bahnoverride für Synchronaktionen: Multiplikative Overridekomponente, wirkt zusätzlich zu Bedien–OVR, programmierten OVR und transformatorischen OVR. DerGesamtfaktor bleibt jedoch begrenzt auf 200%. Muß in jedem Interpolationstakt neu geschrieben werden, sonst wirkt der Wert 100%. Mit $AA_OVR[S1] wird der Spindeloverride verändert. Der durch die Maschinendaten<br>MD 12100: OVR_FACTOR_LIMIT_BIN,<br>MD 12030: OVR_FACTOR_FEEDRATE[30],<br>MD 12010: OVR_FACTOR_AX_SPEED[30],<br>festgelegte Override wird nicht überschritten | | R/W | r/w /4 |
| **$AC_VC** | DOUBLE | Additive Bahnvorschubkorrektur für Synchronaktionen. Der Korrekturwert wirkt nicht bei G0, G33, G331, G332 und G63. Der Korrekturwert muß in jedem Ipotakt neu geschrieben werden, sonst wirkt der Wert 0. Bei Override 0 wird der Korrekturwert unwirksam, sonst wirkt der Override nicht auf den Korrekturwert. Der Gesamtvorschub kann durch den Korrekturwert nicht negativ werden. Nach oben wird so begrenzt, daß die maximalen Achsgeschwindigkeiten und Beschleunigungen nicht überschritten werden. Die Berechnung der anderen Vorschubkomponenten wird nicht durch $AC_VC beeinflußt. Die durch die Maschinendaten<br>MD 12100: OVR_FACTOR_LIMIT_BIN,<br>MD 12030: OVR_FACTOR_FEEDRATE[30],<br>MD 12010: OVR_FACTOR_AX_SPEED[30],<br>MD 12070: OVR_FACTOR_SPIND_SPEED<br>festgelegten Overridewerte werden nicht überschritten. Die additive Vorschubkorrektur wird so begrenzt, daß der resultierende Vorschub den maximalen Overridewert des programmierten Vorschubs nicht überschreitet. | | R/W | r/w /4 |
| **$AC_VACTB** | DOUBLE | Bahngeschwindigkeit im Basiskoordinatensystem | | R | r |
| **$AC_VACTW** | DOUBLE | Bahngeschwindigkeit im Werkstückkoordinatensystem | | R | r |

## Spindeldaten

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AA_S[n]** | DOUBLE /4 | Spindel–Istdrehzahl<br>Das Vorzeichen entspricht der Drehrichtung<br>n: Spindelnummer, 0 ... max. Spindelnummer | Spindel Nr. | RS | r /4 |
| **$AC_CONSTCUT_S[n]** | DOUBLE /6 | Aktuelle konstante Schnittgeschwindigkeit.<br>n: Spindelnummer, 0 ... max. Spindelnummer | Spindel Nr. | RS | r /6 |

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_SDIR[n]** | INT /3 | zur Zeit aktive Spindeldrehrichtung<br>3: Spindeldrehrichtung rechts,<br>4: Spindeldrehrichtung links,<br>5: Spindel Halt<br>n: Spindelnummer, 0 ... max. Spindelnummer | Spindel Nr. | RS | r /3 |
| **$AC_SMODE[n]** | INT /3 | zur Zeit aktive Spindelbetriebsart:<br>0: keine Spindel im Kanal vorhanden<br>1: Drehzahlsteuerbetrieb<br>2: Positionierbetrieb<br>3: Synchronbetrieb<br>4: Achsbetrieb<br>n: Spindelnummer, 0 ... max. Spindelnummer | Spindel Nr. | RS | r /3 |
| **$AC_SGEAR[n]** | INT /5 | zur Zeit aktive Getriebestufe<br>1: 1. Getriebestufe ist aktiv<br>2: 2. Getriebestufe ist aktiv<br>3: 3. Getriebestufe ist aktiv<br>4: 4. Getriebestufe ist aktiv<br>5: 5. Getriebestufe ist aktiv<br>n: Spindelnummer, 0 ... max. Spindelnummer | Spindel Nr. | RS | r /5 |
| **$AC_MSNUM** | INT /3 | Gibt die Nummer der aktuellen Masterspindel zurück:<br>0: keine Spindel vorhanden<br>1..n: Nummer der Masterspindel | | RS | r /3 |
| **$AC_MTHNUM** | INT /5 | Gibt die Nummer der aktuellen Master–WZ–Halter zurück:<br>0: kein Master–WZ–Halter vorhanden<br>1..n: Nummer des Master–WZ–Halter | | RS | r /5 |

### Polynomwerte für Synchronaktionen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_FCTiLL,**<br>**$AC_FCTLL[j]** | DOUBLE<br>/4 | Untere Grenze des Polynoms für Synchronaktionen (SYNFCT)<br>i: 1–3, Auswertefunktion FCTDEF 1 – 3<br>j: Polinomnummer | | R/W | r/w /4 |
| **$AC_FCTiUL,**<br>**$AC_FCTUL[j]** | DOUBLE<br>/4 | Oberer Grenze des Polynoms für Synchronaktionen (SYNFCT)<br>i: 1–3, Auswertefunktion FCTDEF 1 – 3<br>j: Polinomnummer | | R/W | r/w /4 |
| **$AC_FCTiC[n],**<br>**$AC_FCT0[n]** | DOUBLE /(4) | i: 1 – 3, Polynome 1 bis 3; Koeffizienten n: 0 – 3<br>$a_0$–Koeffizient für Polynom n | | R/W | r/w /4 |
| **$AC_FCT1[n]** | DOUBLE /(4) | $a_1$–Koeffizient für Polynom n | | R/W | r/w /4 |
| **$AC_FCT2[n]** | DOUBLE /(4) | $a_2$–Koeffizient für Polynom n | | R/W | r/w /4 |
| **$AC_FCT3[n]** | DOUBLE /(4) | $a_3$–Koeffizient für Polynom n | | R/W | r/w /4 |

## Kanalzustände

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_ALARM_STAT** | INT /5 | (Ausgewählte) Alarmreaktionen für Synchronaktionen (SYNFCT)<br>Bit 2 = 1 NOREADY (aktive Schnellbremsung und Wegnahme der Reglerfreigabe)<br>Bit 6 = 1 STOPBYALARM ( Rampenstop aller Kanal–Achsen)<br>Bit 9 = 1 SETVDI (VDI Nahtstellensignal Alarm wird gesetzt )<br>Bit 13 = 1 FOLLOWUPBYALARM (Nachführen) | – | R | r /5 |
| **$AN_ESR_TRIGGER** | BOOL /5 | $AN_ESR_TRIGGER = 1 Auslösung des ”Erweiterten Stillsetzen und Rückziehens” | – | R/W | r/w /5 |
| **$AC_ESR_TRIGGER** | BOOL /5 | $AC_ESR_TRIGGER = 1<br>Auslösung des ”NC–geführten ESR” | – | R/W | r/w /5 |
| **$AC_OPERATING_ TIME** | DOUBLE /5 | IF $AC_OPERATING_TIME < 12000 GOTOB STARTMARK<br>Gesamt–Laufzeit von NC–Programmen in der Betreibsart Automatik (in Sekunden) | – | R | r /5 |
| **$AC_CYCLE_TIME** | DOUBLE /5 | IF $AC_CYCLE_TIME > 2400 GOTOF ALARM01<br>Laufzeit des angewählten NC–Programms (in Sekunden) | – | R | r /5 |
| **$AC_CUTTING_TIME** | DOUBLE /5 | IF $AC_CUTTING_TIME > 6000 GOTOF ACT_M06<br>Werkzeug–Eingriffszeit (in Sekunden) | – | R | r /5 |
| **$AC_REQUIRED_ PARTS** | DOUBLE /5 | $AC_REQUIRED_PARTS = ACTUAL_LOS<br>Definition der Anzahl benötigter Werkstücke (Werkstück–Soll), z.B. für Definition einer Losgröße, einer Tagesproduktion ... | – | R/W | r/w /5 |
| **$AC_TOTAL_PARTS** | DOUBLE /5 | IF $AC_TOTAL_PARTS > SERVICE_COUNT GOTOF MARK_END<br>Gesamtzeit aller hergestellten Werkstücke | – | R/W | r/w /5 |
| **$AC_ACTUAL_PARTS** | DOUBLE /5 | IF $AC_ACTUAL_PARTS == 0 GOTOF NEW_RUN<br>Anzahl der aktiv hergestellten Werkstücke (Werkstück–Ist). Bei $AC_ACTUAL_PARTS == $AC_REQUIRED_PARTS wird automatisch $AC_ACTUAL_PARTS = 0. | – | R/W | r/w /5 |
| **$AC_SPECIAL_PARTS** | DOUBLE /5 | $AC_SPECIAL_PARTS = R20<br>Anzahl der nach Anwender–Strategie gezählten Werkstücke. Ohne interne Beeinflussung. | – | R/W | r/w /5 |
| **$AC_G0MODE** | INT /6 | Interpolationsverhalten bei G0 Mode<br>0: G0 aktiv<br>1: G0 und Lineare Interpolation aktiv<br>2: G0 und Nicht–Lineare Interpolation aktiv<br>Das Verhalten der Bahnachse bei G0 ist vom Maschinendatum MD 20730: G0_LINEAR_MODE (Siemens–Mode) bzw. Maschinendatum MD 20732: EXTERN_G0_LINEAR_MODE (ISO–Mode) abhängig.<br>Bei Linear Interpolation verfahren die Bahnachsen gemeinsam.<br>Bei Nicht–Linear Interpolation werden die Bahnachsen als Positionierachsen verfahren. | – | R | r /6.1 |
| **$AC_MEAS_LATCH** | DOUBLE /6 | $AC_MEAS_LATCH[0] = 1<br>1. Messpunkt mit Achsistwerten beschreiben.<br>0: entspricht 1. Messpunkt, .. , 3: 4. Messpunkt | Mess–punkt | R/W | r/w /6.1 |

### Positionen

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_IW[X]** | DOUBLE | Istwert im Werkstückkoordinatensystem (WKS) | Achse | R | r |
| **$AA_IEN[X]** | DOUBLE /5 | Istwert im einstellbaren Nullpunkt–Koordinaten-system (ENS). | Achse | R | r /5 |
| **$AA_IBN[X]** | DOUBLE /5 | Istwert im Basis–Nullpunkt–Koordinatensystem (BNS). | | R | r /5 |
| **$AA_IB[X]** | DOUBLE | Istwert im Basiskoordinatensystem (BKS) | Achse | R | r |
| **$AA_IM[X]** | DOUBLE | Istwert im Maschinenkoordinatensystem (MKS) | Achse | R | r |

### Teilungsachsen

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_ACT_IN-DEX_AX_POS_NO[X]** | INT /5 | 0: keine Teilungsachse, damit keine Teilung-sposition verfügbar.<br>> 0: Nummer der zuletzt erreichten bzw. über-fahrenen Teilungsposition | | R | r /5 |
| **$AA_PROG_IN-DEX_AX_POS_NO[X]** | INT /5 | 0: keine Teilungsachse, damit keine Teilung-sposition verfügbar oder<br>die Teilungsachse fährt derzeit keine Teilung-sposition an<br>> 0: Nummer der programmierten Teilungsposi-tion | | R | r /5 |

### Gebergrenzfrequenz

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_ENC_ACTIVE[X]** | BOOL /4 | Aktives Meßsystem arbeitet unterhalb der Ge-bergrenzfrequenz (gültige Werte) | Achse | R | r /4 |
| **$AA_ENCi_ACTIVE[X]** | BOOL /4 | i: 1 – 2 Gebernummer;<br>Meßsystem i arbeitet unterhalb der Gebergrenz-frequenz (gültige Werte) | Achse | R | r /4 |

## Geberwerte

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$VA_IM[X]** | DOUBLE /4 | Endocder–Istwert im Maschinenkoordinatensystem (gemessen aktives Meßsystem), Istwert–Kompensationen sind korrigiert (Spindelsteigungs–Fehlerkompensation, Losekompensation, Quadrantenfehlerkompensation). Eine Modulo–Wandlung findet nicht statt. | Achse | R | r /4 |
| **$VA_IM1[X]** | DOUBLE /4 | Istwert im Maschinenkoordinatensystem (gemessen Encoder 1), Kompensationen sind korrigiert | Achse | R | r /4 |
| **$VA_IM2[X]** | DOUBLE /4 | Istwert im Maschinenkoordinatensystem (gemessen Encoder 2), Kompensationen sind korrigiert | Achse | R | r /4 |
| **$AA_MW[X]** | DOUBLE | Meßwert im Werkstückkoordinatensystem | Achse | R/W | r/w |
| **$AA_MM[X]** | DOUBLE | Meßwert im Maschinenkoordinatensystem | Achse | R/W | r/w /4 |
| **$AA_MWi[X]** | DOUBLE /4 | Meßergebnis achsiales Messen<br>i: 1 –4 für Triggerereignis 1 – 4 | Achse | R/W | r/w |

## Axiales Messen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AA_MMi[X]** | DOUBLE /4 | Meßergebnis achsiales Messen<br>i: 1 –4 für Triggerereignis 1 – 4 | Achse | R/W | r/w |
| **$AA_MEAACT[X]** | BOOL /4 | Wert ist genau dann TRUE (1), wenn axiales Messen aktiv für Achse X | Achse | R | r /4 |

## Verschiebungen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AC_DRF[X]** | DOUBLE | DRF–Verschiebung | Achse | R | r |
| **$AC_PRESET[X]** | DOUBLE | Letzter vorgergegebener Preset–Wert | Achse | R | r |
| **$AA_MEAS_P1_ VALID[X]** | INT /6 | $AA_MEAS_P1_VALID[X] = 1<br>1. Messpunkt mit Achsistwert beschreiben | Achse | R/W | r/w /6.1 |
| **$AA_MEAS_P2_ VALID[X]** | INT /6 | $AA_MEAS_P2_VALID[X] = 1<br>2. Messpunkt mit Achsistwert beschreiben | Achse | R/W | r/w /6.1 |
| **$AA_MEAS_P3_ VALID[X]** | INT /6 | $AA_MEAS_P3_VALID[X] = 1<br>3. Messpunkt mit Achsistwert beschreiben | Achse | R/W | r/w /6.1 |
| **$AA_MEAS_P4_ VALID[X]** | INT /6 | $AA_MEAS_P4_VALID[X] = 1<br>4. Messpunkt mit Achsistwert beschreiben | Achse | R/W | r/w /6.1 |
| **$AA_OFF[X]** | DOUBLE | Überlagerte Bewegung für die programmierte Achse | Achse | R/W | r/w |
| **$AA_OFF_LIMIT[X]** | INT /4 | Grenzwert für die achsiale Korrektur $AA_OFF[X]<br>0: Grenzwert nicht erreicht<br>1: Grenzwert in positiver Achsrichtung erreicht<br>–1: Grenzwert in negativer Achsrichtung erreicht | Achse | R | r |
| **$AA_OFF_VAL[X]** | DOUBLE | $AA_OFF_VAL[X]<br>Integrierter Wert der überlagerten Bewegung für eine Achse. Eine überlagerte Bewegung kann mit Hilfe des negativen Wertes dieser Variable wieder rückgängig gemacht werden. z.B. $AA_OFF[Achse] = –$AA_OFF_VAL[Achse] | Achse | R | r /5 |
| **$AC_RETPOINT[X]** | DOUBLE | Rücksetzpunkt an der Kontur für das Wiederanfahren | Achse | R | r /4 |
| **$AA_SOFTENDP[X]** | DOUBLE | Software–Endlage, positive Richtung | Achse | R | r /4 |
| **$AA_SOFTENDN[X]** | DOUBLE | Software–Endlage, negative Richtung | Achse | R | r /4 |

## Axiale Wege

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AA_DTBW[X]** | DOUBLE | axialer Weg vom Satzanfang im Werkstückkoordinatensystem für Positionier– und Synchronachsen bei Bewegungssynchronaktionen. Für die Wegberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | Achse | R | r |
| **$AA_DTBB** | DOUBLE | axialer Weg vom Satzanfang im Basiskoordinatensystem für Positionier– und Synchronachsen bei Bewegungssynchronaktionen. Für die Wegberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | Achse | R | r |
| **$AA_DTEW** | DOUBLE | axialer Weg zum Satzende im Werkstückkoordinatensystem für Positionier– und Synchronachsen bei Bewegungssynchronaktionen. Für die Wegberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | Achse | R | r |
| **$AA_DTEB** | DOUBLE | axialer Weg zum Satzende im Basiskoordinatensystem für Positionier– und Synchronachsen bei Bewegungssynchronaktionen. Für die Wegberechnung ist allein die programmierte Position maßgeblich. Falls die Achse Koppelachse ist, bleibt der aus der Achskopplung resultierende Positionsanteil hier unberücksichtigt. | Achse | R | r |

## Pendeln

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AA_DTEPW** | DOUBLE | axialer Restweg für Zustellung Pendeln im Werkstückkoordinatensystem | Achse | R | r |
| **$AA_DTEPB** | DOUBLE | axialer Restweg für Zustellung Pendeln im Basiskoordinatensystem | Achse | R | r |
| **$AA_OSCILL_REVERSE_POS1[X]** | DOUBLE | aktuelle Umkehrposition 1 für Pendeln<br>In Synchronaktionen wird der Settingdatenwert $SA_OSCILL_REVERSE_POS1 online ausgewertet | Achse | R | r |
| **$AA_OSCILL_REVERSE_POS2[X]** | DOUBLE | aktuelle Umkehrposition 2 für Pendeln<br>In Synchronaktionen wird der Settingdatenwert $SA_OSCILL_REVERSE_POS2 online ausgewertet | Achse | R | r |

| | | | | | |
|---|---|---|---|---|---|
| **$AA_DELT** | DOUBLE | axialer Restweg im Werkstückkoordinatensystem nach axialem Restweglöschen bei Bewegungssynchronaktionen | Achse | R | r |

### Geschwindigkeiten achsspezifisch

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_OVR[X]** | DOUBLE | axialer Override für Bewegungssynchronaktionen. Multiplikative Overridekomponente, wirkt zusätzlich zu Bedien–OVR, programmierten OVR und transformatorischen OVR, derGesamtfaktor bleibt jedoch begrenzt auf max. 200%. Muß in jedem Ipo–Takt neu geschrieben werden, sonst wirkt der Wert 100%. Mit $AA_OVR[S1] wird der Spindeloverride verändert. Der durch die Maschinendaten MD 12100: OVR_FACTOR_LIMIT_BIN, MD 12030: OVR_FACTOR_FEEDRATE[30], MD 12010: OVR_FACTOR_AX_SPEED[30], $AA_OVR_FACTOR_SPIND_SPEED festgelegte Override wird nicht überschritten. | Achse | R/W | r/w /4 |
| **$AA_VC[X]** | DOUBLE | Additive achsiale Vorschubkorrektur für Bewegungssynchronaktionen. Der Korrekturwert muß in jedem Ipo–Takt neu geschrieben werden, sonst wirkt der Wert 0. Bei Override 0 wird der Korrekturwert unwirksam, sonst wirkt der Override nicht auf den Korrekturwert. Der Gesamtvorschub kann durch den Korrekturwert nicht negativ werden. Nach oben wird so begrenzt, daß die maximalen Achsgeschwindigkeiten und Beschleunigungen nicht überschritten werden. Die Berechnung der anderen Vorschubkomponenten wird nicht durch $AA_VC beeinflußt. Die durch die Maschinendaten MD 12100: OVR_FACTOR_LIMIT_BIN, MD 12030: OVR_FACTOR_FEEDRATE[30], MD 12010: OVR_FACTOR_AX_SPEED[30], MD 12070: OVR_FACTOR_SPIND_SPEED festgelegte Overridewerte werden nicht überschritten . Die additive Vorschubkorrektur wird so begrenzt, daß der resultierende Vorschub den maximalen Overridewert des programmierten Vorschubs nicht überschreitet. | Achse | R/W | r/w /4 |
| **$AA_VACTB[X]** | DOUBLE | Achsgeschwindigkeit im Basiskoordinatensystem Die Variable ist nur aus den Synchronaktionen zugreifbar. | Achse | R | r |
| **$AA_VACTW[X]** | DOUBLE | Achsgeschwindigkeit im Werkstückkoordinatensystem Die Variable ist nur aus den Synchronaktionen zugreifbar. | Achse | R | r |
| **$AA_VACTM[X]** | DOUBLE | Achsgeschwindigkeit sollwertseitig im Maschinenkoordinatensystem. Lesbar auch für Tausch– und PLC–Achsen Die Variable ist nur aus den Synchronaktionen zugreifbar. | Achse | R | r /4 |
| **$VA_VACTM[X]** | DOUBLE | Achsgeschwindigkeit istwertseitig im Maschinenkoordinatensystem. Die Variable liefert einen undefinierten Wert, wenn die Gebergrenzfrequenz überschritten ist. | Achse | R | r /4 |

### Antriebsdaten

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_LOAD[X]** | DOUBLE | Antriebsauslastung in % (nur bei 611D) | Achse | R | r |
| **$VA_LOAD[X]** | DOUBLE | Antriebsauslastung in % (nur bei 611D) | Achse | R | r /5 |
| **$AA_TORQUE[X]** | DOUBLE | Antriebsmomentensollwert in Nm (nur bei 611D)<br>Kraftistwert in N (nur bei 611D–HLA) | Achse | R | r |
| **$VA_TORQUE[X]** | DOUBLE | Antriebsmomentensollwert in Nm (nur bei 611D)<br>Kraftistwert in N (nur bei 611D–HLA) | Achse | R | r /5 |
| **$AA_POWER[X]** | DOUBLE | Antriebswirkleistung in W (nur bei 611D) | Achse | R | r |
| **$VA_POWER[X]** | DOUBLE | Antriebswirkleistung in W (nur bei 611D) | Achse | R | r /5 |
| **$AA_CURR[X]** | DOUBLE | Stromistwert der Achse oder Spindel<br>(nur bei 611D) | Achse | R | r |
| **$VA_CURR[X]** | DOUBLE | Stromistwert der Achse oder Spindel<br>(nur bei 611D) | Achse | R | r /5 |
| **$VA_VAVELIFT[X]** | DOUBLE | Ventilisthub in mm (nur bei 611D–Hydraulik) | Achse | R | r /5 |
| **$VA_PRESSURE_A[X]** | DOUBLE | Druck auf A–Seite des Zylinders in bar<br>(nur bei 611D–Hydraulik) | Achse | R | r /5 |
| **$VA_PRESSURE_B[X]** | DOUBLE | Druck auf B–Seite des Zylinders in bar<br>(nur bei 611D–Hydraulik) | Achse | R | r /5 |

### Achszustände

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_STAT[X]** | INT<br>/4 | Achsstatus:<br>0: Kein Achsstatus verfügbar<br>1: Verfahrbewegung steht an<br>2: Achse hat Ipo–Ende erreicht, nur für Achsen des Kanals<br>3: Achse in Position (Genauhalt Grob) für alle Achsen<br>4: Achse in Position (Genauhalt Fein) für alle Achsen | Achse | R | r /4 |
| **$AA_REF[X]** | INT<br>/5 | Achsstatus:<br>0: Achse ist nicht referiert<br>1: Achse ist referiert | | R | r /5 |
| **$AA_TYP[X]** | INT<br>/4 | AchsTyp:<br>0: Achse in anderem Kanal<br>1: Kanalachse des eigenen Kanals<br>2: neutrale Achse<br>3: PLC–Achse<br>4: Pendelachse<br>5: neutrale Achse, die akt. in JOG verf. wird<br>6: Leitwertgekoppelte Folgeachse<br>7: Mitschleppen Folgeachse<br>8: Kommandoachse<br>9: Compile–Cyclen–Achse | Achse | R | r /4 |
| **$AA_MASL_STAT[X]** | INT<br>6 | Aktueller Zustand einer Master–Slave Kopplung $AA_MASL_STAT auf Wert<br>= 0: Achse ist keine Slaveachse bzw. keine Kopplung ist aktiv.<br>> 0: Kopplung aktiv, liefert die zugehörige Maschinenachsnummer der Masterachse. | Slave-Achse | R | r /6 |

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_FXS[X]** | INT<br>5 | Status ”Fahren auf Festanschlag”<br>0: Achse nicht im Anschlag<br>1: Anschlag wurde erfolgreich angefahren<br>2: Anfahren des Festanschlags fehlgeschlagen<br>3: Anwahl Fahren auf Festanschlag aktiv<br>4: Anschlag wurde erkannt<br>5: Abwahl Fahren auf Festanschlag aktiv | Achse | R/W | r /5 |
| **$VA_TORQUE_AT_ LIMIT[X]** | INT<br>5 | Status ”Wirksames Moment entspricht der vorgegebenen Momentgrenze”<br>0:Momentengrenze noch nicht erreicht<br>1:Momentengrenze erreicht<br>In digitalen Systemen 611D wird vom Antrieb der Status geliefert, ob die programmierte Momentengrenze erreicht ist. | Achse | R | r /5 |
| **$AA_FOC[X]** | INT<br>5 | Status der Funktion ”ForceControl” (FOC)<br>0: FOC nicht aktiv<br>1: FOC modal aktiv<br>2: FOC satzbezogen aktiv | Achse | R/W | r /5 |
| **$AA_COUP_ ACT[SPI(2)]** | INT<br>/4 | Aktueller Kopplungszustand Folgespindel/Achse<br>0: Achse/Sp. hat keine Kopplung an eine LS/LA<br>3: Achse wird tangentiell nachgeführt<br>4: Synchronspindelkopplung<br>8: Achse wird mitgeschleppt<br>16: Folgeachse der Leitwertkopplung<br>Die jeweiligen Werte gelten für eine Kopplung. Sind mehrere Kopplungen für die Folgeachse aktiv, dann wird dies durch die Summe der jeweiligen Zahlenwerte repräsentiert. | Folge–spindel/ Achse | R | r /5 |

## Elektronisches Getriebe 1

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_EG_SYNFA[a]** | DOUBLE<br>/5 | Synchronposition der Folgeachse<br>a: Folgeachse | Fol-geachse | R | r /5 |
| **$AA_EG_NUM_LA[a]** | INT<br>/5 | Anzahl der mit EGDEF spezifizierten Leitachsen<br>a: Folgeachse | Fol-geachse | R | r /5 |
| **$AA_EG_SYNCDIFF[a]** | DOUBLE<br>/5 | Synchronlaufdifferenz<br>a: Folgeachse | Fol-geachse | R | r /5 |
| **$AA_EG_AX[n,a]** | AXIS<br>/5 | Bezeichner für die n–te Leitachse<br>n: Index für Leitachse<br>a: Folgeachse | Fol-geachse | R | r /5 |

## Leitwertkopplung

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$AA_LEAD_SP[LW]** | DOUBLE /4 | simulierterer Leitwert – Position | Leitwert | R/W | r/w |
| **$AA_LEAD_SV[LW]** | DOUBLE /4 | simulierterer Leitwert – Geschwindigkeit | Leitwert | R/W | r/w |
| **$AA_LEAD_P_ TURN[LW]** | DOUBLE /4 | aktueller Leitwert – Positionsanteil, der durch Moduloreduktion verlorengeht. Die tatsächliche Leitwert–Position (mit der die Steuerung intern rechnet) ist $AA_LEAD_P[LW] + $AA_LEAD_P_TURN[LW]. Wenn LW eine Moduloachse ist, ist $AA_LEAD_P_TURN ein ganzzahliges Vielfaches von $MA_MODULO_RANGE. Wenn LW keine Moduloachse ist, ist $AA_LEAD_P_TURN stets 0.<br>Beispiel1:<br>$MA_MODULO_RANGE[LW] = 360<br>$AA_LEAD_P[LW] = 290<br>$AA_LEAD_P_TURN[LW] = 720<br>Die tatsächliche Leitwert–Position (mit der die Steuerung intern rechnet) ist 1010.<br>Beispiel_2:<br>$MA_MODULO_RANGE[LW] =360<br>$AA_LEAD_P[LW] = 290<br>$AA_LEAD_P_TURN[LW] =–360<br>Die tatsächliche Leitwert–Position (mit der die Steuerung intern rechnet) ist –70. | Leitwert | R | r /4 |
| **$AA_LEAD_P[LW]** | DOUBLE /4 | aktueller Leitwert – Position (modulo reduziert)<br>Wenn LW eine Moduloachse ist, gilt stets<br>0 <= $AA_LEAD_P[LW] <= $MA_MO-DULO_RANGE[LW] | Leitwert | R | r /4 |
| **$AA_LEAD_V[LW]** | DOUBLE /4 | aktueller Leitwert – Geschwindigkeit | Leitwert | R | r /4 |
| **$AA_SYNC[FA]** | INT /4 | Kopplungszustand der Folgeachse bei Leitwert-kopplung<br>0: keine Synchronität<br>1: Synchronlauf Grob<br>2: Synchronlauf Fein<br>3: Grob und Fein | Folge-achse | R | r /4 |

## Synchronspindel

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$P_COUP_OFFS[S2]** | DOUBLE | Programmierter Offset der Synchronspindel | Folges-pindel | R | r /6 |
| **$AA_COUP_OFFS[S2]** | DOUBLE /2 | Positionsoffset für Synchronspindel sollwertsei-tig | Folges-pindel | R | r /4 |
| **$VA_COUP_ OFFS[SPI(2)]** | DOUBLE /2 | Positionsoffset für Synchronspindel istwertseitig | Folges-pindel | R | r /4 |

### Safety Integrated 1

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$VA_IS[X]** | DOUBLE /3 | sichere Istposition (SISITEC) | Achse | R | r /4 |
| **$AA_SCTRACE[X]** | BOOL /4 | $AA_SCTRACE[X] = 1<br>Schreiben:<br>IPO–Trigger für Servo–Trace auslösen.<br>0: keine Aktion<br>1: Trigger auslösen<br>Lesen: stets Wert 0, da das selbslöschend Triggerbit aus der Schnittstelle zurückgelesen wird.<br>0: aktueller Wert (kein Status) | Achse | R/W | r /4 |
| **$VA_DPE[X1]** | BOOL /5 | Status der Leistungsfreigabe einer Maschinenachse (611D und 611D–Hydraulik)<br>FALSE: keine Leistungsfreigabe<br>TRUE: Leistungsfreigabe vorhanden | Masch.–Achse | R/ | r /5 |
| **$AA_ACC** | DOUBLE /5 | Aktuellen Beschleunigungswert der Achse bei Einachsinterpolation.<br>$AA_ACC = $MA_MAX_AX_ACCEL*progr. Beschleunigungskorrektur | Achse | R/ | r /5 |
| **$AA_MOTEND** | INT /5 | Aktuelles Bewegungsendekriterium b. Einachs–interpolation.<br>1: Bewegungsende bei Genauhalt FEIN<br>2: Bewegungsende bei Genauhalt GROB<br>3: Bewegungsende bei Genauhalt IPO–Stop<br>4: Satzwechsel in der Bremsrampe der Achsbewegung | Achse | R/ | r /5<br>r /6 |
| **$AA_SCPAR** | INT /5 | Aktuellen Servo Parametersatz lesen | Achse | R/ | r /5 |

### Erweitertes Stillsetzen–Rückziehen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AA_ESR_STAT[X]** | INT /5 | Status des ”Erweiterten Stillsetzen und Rückziehens”, bitcodiert:<br>BIT0: Generatorbetrieb ist ausgelöst<br>BIT1: Rueckziehen ist ausgelöst<br>BIT2: Erw. Stillsetzen ist ausgelöst<br>BIT3: Zwischenkreis–Unterspannung<br>BIT4: Generator–Minimaldrehzahl | | R/ | r/ |
| **$AA_ESR_ENABLE[X]** | BOOL /5 | $AA_ESR_ENABLE[X] = 1<br>Freigabe des ”Erweiterten Stillsetzen und Rückziehens” | | R/W | r/w |

## Achscontainer–Drehung

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AN_AXCTSWA[n]** | BOOL /5 | Achscontainerdrehung aktiv<br>1: eine Achs–Container–Drehung wird augenblicklich auf dem Achs–Container mir dem Achs–Containernamen n ausgeführt<br>0: es ist keine Achs–Container–Drehung aktiv | Achscontainer | r/ | r/ |
| **$AN_AXCTAS[n]** | INT /5 | Achs–Container–Drehung, aktuelle Verdrehung:<br><br>Für den Achs–Container mit dem Achs–Container–Namen n wird angegeben,<br>um wieviele Plätze (Slots) der Achs–Container aktuell weitergeschaltet wurde.<br>Der Wertebereich liegt von 0 bis maximale Anzahl belegter Plätze im Achs–Container –1 | Achscontainer | r/ | r/ |
| **$AC_AXCTSWA[n]** | BOOL /5 | Achscontainerdrehung freigeben<br>1: Der Kanal hat für den Achs–Container–Namen n die Achs–Container–Drehung freigegeben und diese ist noch nicht beendet.<br>0: Die Achs–Container–Drehung ist beendet. | Achscontainer | r/ | r/ |

## Elektronisches Getriebe 2

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$AA_EG_TYPE** | INT /5 | $AA_EG_TYPE[a,b]<br>a: Folgeachse<br>b: Leitachse<br>Art der Kopplung für die Leitachse b<br>0: Istwertkopplung<br>1: Sollwertkopplung | Achse | R | r /5 |
| **$AA_EG_NUMERA** | DOUBLE /5 | $AA_EG_NUMERA[a,b]<br>a: Folgeachse<br>b: Leitachse<br>Zähler des Koppelfaktors für die Leitachse b | Achse | R | r /5 |
| **$AA_EG_DEMOM** | DOUBLE /5 | $AA_EG_DENOM[a,b]<br>a: Folgeachse<br>b: Leitachse<br>Nenner des Koppelfaktors für die Leitachse b | Achse | R | r /5 |
| **$AA_EG_SYN** | DOUBLE /5 | $AA_EG_SYN[a,b]<br>a: Folgeachse<br>b: Leitachse<br>Synchronposition der Leitachse b | Achse | R | r /5 |
| **$AA_EG_AKTIVE** | BOOL /5 | $AA_EG_AKTIVE[a,b]<br>a: Folgeachse<br>b: Leitachse<br>Kupplung für die Leitachse b ist aktiv. d.h. eingeschaltet | Achse | R | r /5 |

### Safety Integrated (S. I. )

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_INSE[n]** | BOOL /4 | Abbild eines Safety–Eingangs–Signals (ext. NCK–Schnittstelle) | Nr. Ein-gang | R | r |
| **$A_INSED[n]** | INT /5 | Abbild der Safety–Eingangs–Signale (ext. NCK–Schnittstelle) | Nummer des Ein-gangs–Wortes 1 – ... | R/ | r/ |
| **$A_INSEP[n]** | BOOL /5 | Abbild eines Safety–Eingangs–Signal (ext.PLC–Schnittstelle) | Nummer des Ein-gangs 1 – ... | R/ | r/ |
| **$A_INSEPD[n]** | INT /5 | Abbild der Safety–Eingangs–Signale (ext. PLC–Schnittstelle) | Nummer des Ein-gangs–Wortes 0 – ... | R/ | r/ |
| **$A_OUTSE[n]** | BOOL /5 | Abbild eines Safety–Ausgangs–Signal (ext. NCK–Schnittstelle) | Nummer des Ausgang 1 – ... | R/W | r/w |
| **$A_OUTSED[n]** | INT /5 | Abbild der Safety–Ausgangs–Signale (ext. NCK–Schnittstelle) | Nummer des Aus-gangs–Wortes 1 – ... | R/W | r/w |
| **$A_OUTSEP[n]** | BOOL /4 | Abbild eines Safety–Ausgangs–Signal (ext. PLC–Schnittstelle) | Nummer des Ausgang s 1 – ... | R | r/ |
| **$A_OUTSEPD[n]** | INT /5 | Abbild der Safety–Ausgangs–Signale (ext. PLC–Schnittstelle) | Nummer des Aus-gangs–Wortes 0 – ... | R | r/ |

### S. I. : Servo–Interpolator–Schnittstelle

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_INSI[n]** | BOOL /4 | Abbild eines Safety–EIngangs–Signal (int. NCK–Schnittstelle) | Nummer des Eingang s 1 – ... | R | r/ |
| **$A_INSID[n]** | INT /5 | Abbild der Safety–EIngangs–Signale (int. NCK–Schnittstelle) | Nummer des Eingang s–Worte s 1 – ... | R | r/ |
| **$A_INSIP[n]** | BOOL /4 | Abbild eines Safety–EIngangs–Signal (int. PLC–Schnittstelle) | Nummer des Eingang s–Worte s 1 – ... | R | r/ |

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_INSIPD[n]** | INT /5 | Abbild der Safety–EIngangs–Signale (int. PLC–Schnittstelle) | Nummer des Eingangs–Worte s 1 – ... | R | r/ |
| **$A_OUTSI[n]** | BOOL /4 | Abbild eines Safety–Ausgangs–Signal (int. NCK–Schnittstelle) | Nummer des Ausgangs 1 – ... | R/W | r/w |
| **$A_OUTSID[n]** | INT /5 | Abbild der Safety–Ausgangs–Signale (int. NCK–Schnittstelle) | Nummer des Ausgangs–Worte s 1 – ... | R/W | r/w |
| **$A_OUTSIP[n]** | BOOL /4 | Abbild eines Safety–Ausgangs–Signal (int. PLC–Schnittstelle) | Nummer des Ausgangs 1 – ... | R | r/ |
| **$A_OUTSIPD[n]** | INT /5 | Abbild der Safety–Ausgangs–Signale (int. PLC–Schnittstelle) | Nummer des Ausgangs–Worte s 1 – ... | R | r/ |

### Safety–Merker und Timer

| Name | Typ /SW | Beschreibung/Werte | Index | Zu-griff TP | Zu-griff SA /SW |
|---|---|---|---|---|---|
| **$A_MARKERSI[n]** | BOOL /4 | Merker für Safety–Integrated–Programmierung | Nr. Merkers 1 ... | R/W | r/w |
| **$A_MARKERSID[n]** | INT /5 | Merkerwort (32Bit) für Safety–Programmierung | Nr. des Merkerwort 1 ... | R/W | r/w |
| **$A_MARKERSIP[n]** | BOOL /4 | Abbild der PLC–Safety–Integrated–Merker | Nr. Merkers 1 ... | R | r/ |
| **$A_MARKERSIPD[n]** | INT /5 | Abbild der PLC–Safety–Integrated–Merkerworte | Nr. des Merkerwort 1 ... | R | r/ |
| **$A_TIMERSI[n]** | DOUBLE /4 | Safety–Timer – Einheit in Sekunden.<br>Die Zeit wird intern gerastet in Vielfachen des Interpolationstaktes gezählt;<br>Das Hochzählen der Zeitvariablen wird gestartet durch die Wertzuweisung $A_TIMERSI[n]=<startwert> .<br>Das Hochzählen einer Zeitvariable wird gestoppt durch Zuweisung eines negativ. Wertes: $A_TIMERSI[n]=–1.<br>Der aktuelle Zeitwert kann bei laufender oder gestoppter Zeitvariable gelesen werden. Nach dem Stoppen der Zeitvariable durch Zuweisung von –1 bleibt der zuletzt aktuelle Zeitwert stehen und kann weiterhin gelesen werden. | Nr. des Timers 1 ... | R/W | r/w |

### Safety–KDV–Steuer– und Statusvariablen

| Name | Typ /SW | Beschreibung/Werte | Index | Zugriff TP | Zugriff SA /SW |
|---|---|---|---|---|---|
| **$A_STATSID** | INT /5 | Safety: Status des Kreuzweisen Datenvergleichs (KDV) zwischen NCK und PLC wenn Wert ungleich Null, ist Fehler in KDV aufgetreten | – | R/ | r/ |
| **$A_CMDSI[n]** | BOOL /5 | Safety: Steuerwort für den Kreuzweisen Datenvergleich (KDV) zwischen NCKund PLC. Feldindex n = 1: Zeit für Signal–Änderungsüberwachung auf 10 s erhöhen | Nummer des Steuersignals | R/W | r/w |
| **$A_LEVELSID** | INT /5 | Safety: Anzeige des Füllstands der Signal–Änderungsüberwachung. Gibt die Anzahl der momentan zur Überprüfung durch den Kreuzweisen Datenvergleich vermerkten Signale an. | – | R | r/ |

## 2.4 Aktionen in Synchronaktionen

**Aktionen**

Jede Synchronaktion enthält nach dem Aktionskennwort **DO ...**

– eine oder mehrere (max. 16) Aktionen oder einen Technologiezyklus (Als Oberbegriff wird im Weiteren **Aktionen** verwendet.)

die bei erfüllter Bedingung ausgeführt werden.

**Mehrere Aktionen**

Mehrere Aktionen einer Synchronaktion werden bei erfüllter Bedingung im gleichen Interpolationstakt aktiviert.

**Liste möglicher Aktionen**

Im Aktionsteil von Synchronaktionen sind die folgenden Aktionen möglich:

Tabelle 2-2 Aktionen in Synchronaktionen

| ... **DO** ... | **Bedeutung** | **Verweis** |
|---|---|---|
| Mxx<br>Sxx<br>Hxx | Hilfsfunktionsausgabe an PLC | 2.4.1 |
| SETAL(nr) | Alarm setzen, Reaktion auf Fehler | 2.4.20 |
| $A...= ... $V... = ...<br>$AA_OFF =<br><br>$AC_OVR =<br>$AA_OVR =<br>$AC_VC =<br>$AA_VC =<br>$$SN_SW_CAM_ ...<br><br>$AC_FCT... | Schreiben von Echtzeitvariablen:<br>– Überlagerte Bewegung<br>– Geschwindigkeitsbeeinflussung:<br>Bahngeschwindigkeit<br>Achsgeschwindigkeit<br>add. Bahnvorschubkorrektur<br>add. Korrekturwert der Achse<br>Verändern von SW–Nockenpositionen (Settingdaten) und alle anderen SD<br>Überschreiben von FCTDEF–Parametern | 2.4.2<br><br><br><br><br><br><br>2.4.3<br><br>2.4.4 |
| <br>RDISABLE<br>STOPREOF<br>DELDTG<br>FTOC<br>SYNFCT<br>ZYKL_T1 (z.B.) | Synchronaktionsprozeduren:<br>Einlesesperre aktivieren<br>Vorlaufstop beenden<br>Restweg löschen<br>Online–Werkzeugkorrektur<br>Polynomauswertung<br>Aufruf von Technologiezyklen | <br>2.4.8<br>2.4.9<br>2.4.10<br>2.4.7<br>2.4.5<br>2.5 |
| <br>$AA_OVR[x]= 0<br>ACHSE_X (z.B.)<br>POS[u]= ...<br>FA[u]= ...<br><br>MOV[u]= >0<br>MOV[u] = <0<br>MOV[u] = =0 | Steuerung von Positionierachsen:<br>Sperren einer Achsbewegung<br>Aufruf eines Achsprogrammes<br>Positionieren<br>Achsvorschub festlegen<br>Kommando–Achsen kontinuierl. bewegen:<br>– vorwärts<br>– rückwärts<br>– anhalten | <br>2.4.11<br>2.4.12<br>2.4.12<br>2.4.13<br>2.4.14<br>”<br>”<br>” |
| <br>SPOS<br>M3, M4, M5, S =<br>$AA_OVR[S1]= 0 | Spindeln:<br>Positionieren<br>Drehrichtung, Halt, Drehzahl<br>Sperren der Spindelbewegung | <br>2.4.15 |
| PRESETON( , ) | Istwertsetzen | 2.4.16 |

Tabelle 2-2 Aktionen in Synchronaktionen

| **... DO** ... | **Bedeutung** | **Verweis** |
|---|---|---|
| <br>LEADON<br>LEADOF<br>TRAILON<br>TRAILOF | Kopplungen aktivieren/deaktivieren:<br>Folgeachse an Leitachse ankoppeln<br>Kopplung aufheben<br>Asynchr. Mitschleppen ein<br>Asynchr. Mitschleppen aus | 2.4.17 |
| MEAWA,<br>MEAC | Messen ohne Restweglöschen<br>zyklisches Messen | 2.4.18 |
| <br>SETM<br>CLEARM | Kanalsynchronisation:<br>Setzen einer Wartemarke<br>Löschen einer Wartemarke | 2.4.19 |
| <br>LOCK<br><br>UNLOCK<br><br>RESET | Koordinierung zw. Synchronaktionen:<br>– Synchronaktion / Technologiezyklus sperren<br>– Synchronaktion / Technologiezyklus freigeben<br>– Technologiezyklus rücksetzen | 2.5.1 |

## 2.4.1 Ausgabe von M–, S– und H–Hilfsfunktionen an die PLC

Details zur Hilfsfunktionsausgabe im Allgemeinen finden Sie in:

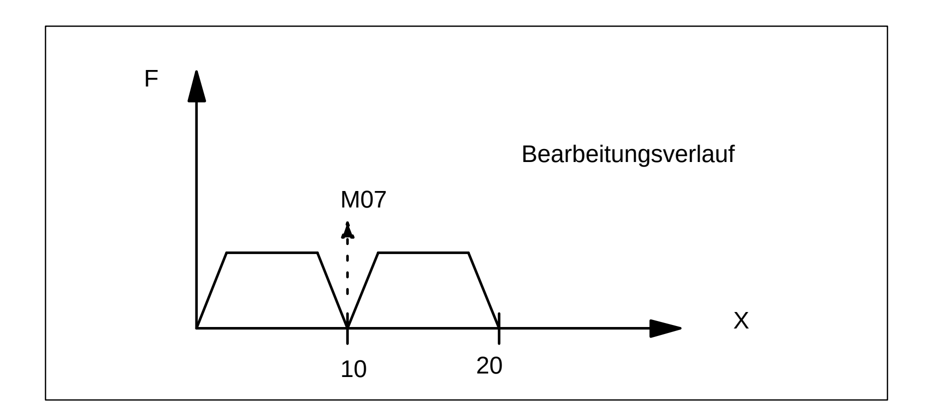
**Literatur:** /FB/, H2, ”Hilfsfunktionsausgabe an PLC”

**Beispiele**

Der Vorteile der Hilfsfunktionsausgabe aus Synchronaktionen wird am folgenden Beispiel deutlich: Kühlmittel an bestimmter Position einschalten

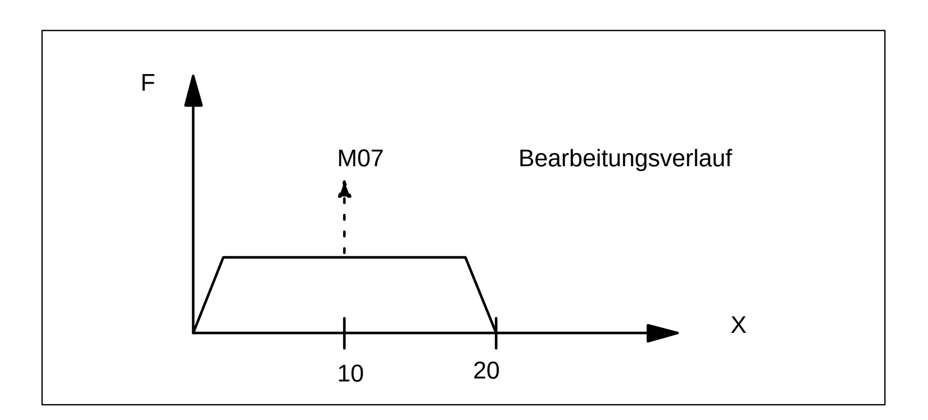
Lösung **ohne** Synchronaktion: 3 Sätze

```
N10 G1 X10 F150
N20 M07
N30 X20
```




Lösung **mit** Synchronaktion: 1 Satz

```
N10 WHEN $AA_IM[X] >= 10 DO M07
N20 G1 X20 F150
```



**Hilfsfunktionsausgabe an die PLC**

Als Synchronaktion können M–, S– oder H–Hilfsfunktionen an die PLC ausgegeben werden. Die Ausgabe erfolgt **sofort** (wie ein Interrupt an PLC) im Interpolationstakt, wenn die Bedingung erfüllt ist.
Der im

MD 11110: AUXFU_GROUP_SPEC (Hilfsfunktionsgruppen–spezifikation) bzw.
AUXFU_M_SYNC_TYPE (Ausgabezeitpunkt der M–Funktionen) /
AUXFU_S_SYNC_TYPE (Ausgabezeitpunkt der S–Funktionen) /
AUXFU_H_SYNC_TYPE (Ausgabezeitpunkt der H–Funktionen) /

ggf. definierte Ausgabezeitpunkt ist unwirksam.

**Programmierung**

Hilfsfunktionen dürfen nur mit den Häufigkeitsschlüsseln WHEN oder EVERY in Synchronaktionen programmiert werden.

**Beispiel**

WHEN $AA_IM[X] > 50 DO H15 S3000 M03
; wenn Istwert der X–Achse größer 50 wird, H15 ausgeben, neue Spindeldrehzahl, neue Drehrichtung einstellen

**Einschränkung**

Es können gleichzeitig (d.h. in einem OB40–Zyklus der PLC)
maximal 10 Hilfsfunktionen ausgegeben werden. Die Summe der Hilfsfunktionsausgaben aus Teileprogrammen und Synchronaktionen darf zu keinem Zeitpunkt mehr als 10 pro Kanal betragen.
Höchste Anzahl Hilfsfunktionen je Synchronaktions–Satz oder Technologiezyklus–Satz:

– 5 M–Funktionen
– 3 S–Funktionen
– 3 H–Funktionen

Vordefinierte M–Funktionen können nicht über Synchronaktionen programmiert werden. Sie werden mit Alarm abgelehnt.
WHEN ... DO M0 ; Alarm
Erlaubt sind jedoch die Spindel–M–Funktionen: M3, M4, M5 und
M17 als Ende für einen Technologiezyklus.

**Quittierung**

Technologiezyklensätze (siehe Kap. 2.5 ) mit Hilfsfunktionsausgaben sind erst dann abgearbeitet, wenn die Quittung aller Hilfsfunktionen des Satzes von PLC erfolgt ist. Die Satzweiterschaltung im Technologiezyklus erfolgt erst dann, wenn alle darin enthaltenen Hilfsfunktionen von PLC quittiert sind.

**SW 5**

Ab SW 5 sind weitere Varianten für das Quittungsverhalten eingeführt worden:

– Hilfsfunktionsausgabe ohne Satzwechselverzögerung
  Schnelle Hilfsfunktionen (QUICK) vorab, als paralleler Prozeß in der PLC, danach Hilfsfunktionsausgabe mit Quittungserwartung.

Der Datentyp für H–Hilfsfunktionen kann zwischen INT und REAL durch den Anwender gewählt werden. Das PLC–Anwenderprogramm muß entsprechend der Festlegung die übergebenen Werte interpretieren. Der INT–Wertebereich für H–Hilfsfunktionen wurde vergrößert auf:–2 147 483 648 bis 2 147 483 647.

**Literatur:** /FB/, H2, Hilfsfunktionsausgabe an PLC für SW 5

## 2.4.2 Setzen (Schreiben) und Lesen von Echtzeitvariablen

**Schreiben**

In Synchronaktionen können die Echtzeitvariablen in Aktionen **geschrieben** werden, die in der Liste in Kap. 2.3.8 mit dem Zugriff "Schreiben aus Synchronaktionen" mit + gekennzeichnet sind.

- Maschinen– und Settingdaten z.B. $$MN_..., $$MC_..., $$MA_... bzw. $$SN_..., $$SC_..., $$SA_...

---

**Hinweis**

Maschinen– und Settingdaten, die online im Hauptlauf geschrieben werden sollen, müssen mit **$$**.._... programmiert werden.

---

**Wirksamkeit**

Aus Synchronaktionen geschriebene Maschinendaten müssen mit Wirksamkeit SOFORT gekennzeichnet sein, andernfalls steht der veränderte Wert für die weitere Bearbeitung noch nicht zur Verfügung. Die Angaben zur Wirksamkeit neuer Maschinendatenwerte nach Änderungen finden Sie in:

**Literatur:** /LIS/, Listen

Beispiele:
... DO **$$MN_MD_FILE_STYLE = 3** ; Maschinendatum setzen
... DO **$$SA_OSCILL_REVERSE_POS1 = 10** ; Settigdatum setzen
... DO **$A_OUT[1]=1** ; Digitalen Ausgang setzen
... DO **$A_OUTA[1]= 25** ; Analogwert ausgeben

**Lesen**

Für Zuweisungen an Echtzeitvariablen und als Eingangsgrößen für Funktionen sowie zur Formulierung von Bedingungen kann auf die Variablen in der Synchronaktionen **lesend** zugegriffen werden, die in der Liste in Kap. 2.3.8 mit dem Zugriff "Lesen aus Synchronaktionen" mit **r** gekennzeichnet sind.

- Maschinendaten, Settingdaten z.B. $$SN_..., $$SC_..., $$SA_...

---

**Hinweis**

Maschinendaten und Settingdaten, die online im Hauptlauf angesprochen werden sollen, müssen mit **$$**.._... programmiert werden, wenn die Variablen sich während der Bearbeitung ändern können. Für Variablen, deren Inhalt sich nicht ändert, ist ein $–Zeichen vor dem Bezeichner hinreichend.

---

Beispiele:
WHEN **$AC_DTEB** < 5 DO ... ; Abstand vom Satzende in Bedingung lesen
DO $R5= **$A_INA[2]** ; Wert des Analogeingangs 2 lesen und Rechenvariable zuweisen

### 2.4.3 Verändern von SW–Nockenpositionen und –zeiten (Settingdaten)

**Einführung**

Mit der Funktion ”Softwarenocken” können positionsabhängige Nockensignale an die PLC oder an die NCK–Peripherie ausgegeben werden.

**Literatur:** /FB/, N3, Softwarenocken, Wegschaltsignale

**Funktion**

Über Synchronaktionen können Nockenpositionen, bei denen die Signalausgänge gesetzt werden, durch Beschreiben bestehender Settingdaten verändert werden. Folgende Settingdaten können über Synchronaktionen verändert werden:

```
$$SN_SW_CAM_MINUS_POS_TAB_1[0..7]  ; Positionen der Minusnocken
$$SN_SW_CAM_MINUS_POS_TAB_2[0..7]  ; Positionen der Minusnocken
$$SN_SW_CAM_PLUS_POS_TAB_1[0..7]   ; Positionen der Plusnocken
$$SN_SW_CAM_PLUS_POS_TAB_2[0..7]   ; Positionen der Plusnocken
```

**Beispiel 1**

Veränderung einer Nockenposition:

```
ID=1 WHEN $AA_IW[x] > 0
          DO $$SN_SW_CAM_MINUS_POS_TAB_1[0] = 50.0
```

Vorhalte– bzw. Verzögerungszeiten können über die folgenden Settingdaten verändert werden:

```
$$SN_SW_CAM_MINUS_TIME_TAB_1[0..7] ; Vorhalte– bzw. Verzögerungszeit
                                     an den Minusnocken
$$SN_SW_CAM_MINUS_TIME_TAB_2[0..7] ; Vorhalte– bzw. Verzögerungszeit
                                     an den Minusnocken
$$SN_SW_CAM_PLUS_TIME_TAB_1[0..7] ; Vorhalte– bzw. Verzögerungszeit
                                     an den Plusnocken
$$SN_SW_CAM_PLUS_TIME_TAB_2[0..7] ; Vorhalte– bzw. Verzögerungszeit
                                     an den Plusnocken
```

**Beispiel 2**

Änderung einer Vorhalte–/Verzögerungszeit:

```
ID=1 WHEN $AA_IW[x] > 0
          DO $$SN_SW_CAM_MINUS_TIME_TAB_1[0] = 1.0
```

---

**Hinweis**

Das Setzen der Softwarenocken über Synchronaktionen darf geschwindigkeitsabhängig nicht unmittelbar vor einer Nocke geschehen, sondern es müssen mindestens noch 2 – 3 Interpolationstakte bis zum Erreichen der Nocke zur Verfügung stehen.

---

## 2.4.4 FCTDEF

**Anwendung**

Die in den folgenden Unterkapitel beschriebenen Aktionen Online–Werkzeugkorrektur FTOC und Polynomauswertung SYNFCT
benötigen die Beschreibung eines Zusammenhanges zwischen einer Eingangsgröße und einer Ausgangsgröße durch ein Polynom. FCTDEF definiert solche Polynome.
Spezielle Beispiele für den Polynomeinsatz für Online–Abrichten einer Schleifscheibe finden Sie unter 2.4.7. Beispiele für lastabhängige Vorschübe und Abstandsregelung über Polynome finden Sie unter 2.4.5.

**Eigenschalten der Polynome**

Die mit FCTDEF definierten Polynome haben die folgenden Eigenschaften:

- Erzeugung durch Aufruf FCTDEF im Teileprogramm
- Die Parameter der definierten Polynome sind Echtzeitvariablen.
- Überschreiben einzelner Parameter der Polynome wie Schreiben Echtzeitvariablen, zulässig im Teileprogramm allgemein und im Aktionsteil der Synchronaktionen. S. 2.4.2 .

---

**Hinweis**

Die Möglichkeit, Gültigkeitsgrenzen und Koeffizienten von bestehenden Polynomen aus Synchronaktionen heraus zu verändern, ist ab SW–Stand 4 vorhanden. Beispiel: WHEN ... DO $AC_FCT1[1]= 0.5

---

**Anzahl Polynome**

Ab SW–Stand 4 kann die Anzahl der Polynome, die gleichzeitig definiert sein können, per
MD 28252 : MM_NUM_FCTDEF_ELEMENTS
vorgegeben werden.

**Satzsynchrone Polynomdefinition**

**FCTDEF**(
Polynom–Nr.,
Untergrenze,
Obergrenze,
a0,
a1,
a2,
a3)

Der Zusammenhang zwischen Ausgangsgröße y und Eingangsgröße x ist wie folgt:

$y= a_0+ a_1x+ a_2x^2 + a_3x^3$

Die in der Funktion angegebenen Parameter werden wie folgt in Systemvariablen abgelegt:

$AC_FCTLL[n]: Untergrenze, n: Polynomnummer
$AC_FCTUL[n]: Obergrenze, n: Polynomnummer
$AC_FCT0[n]: a0–Koeffizient, n: Polynomnummer
$AC_FCT1[n]: a1–Koeffizient, n: Polynomnummer
$AC_FCT2[n]: a2–Koeffizient, n: Polynomnummer
$AC_FCT3[n]: a3–Koeffizient, n: Polynomnummer

In Kenntnis dieses Zusammenhangs können die Polynome auch direkt über die Systenvariablen geschrieben oder verändert werde. Der Gültigkeitsbereich des Polynoms wird durch die Grenzen $AC_FCTLL[n] und $AC_FCTUL[n] festgelegt.

**Aufruf der Polynomaus–wertung**

Gespeicherte Polynome können mit den folgenden Funktionen verwendet werden:

– Online Werkzeugkorrektur, FTOC()

– Polynom–Auswertung, SYNFCT().

**Literatur:** /PG/, Programmieranleitung Grundlagen
/PGA/, Programmieranleitung Arbeitsvorbereitung
/FB/, W4 ”Schleifen”

## 2.4.5 Polynomauswertung SYNFCT

**Anwendung**

Mit einer Auswertefunktion im Aktionsteil der Synchronaktion kann bearbeitungssynchron eine Variable gelesen, mit einem Polynom bewertet und das Ergebnis in eine andere Variable geschrieben werden. Damit können z.B. folgende Aufgabenstellungen gelöst werden:

- Vorschub in Abhängigkeit von der Antriebsauslastung
- Position in Abhängigkeit von einem Sensorsignal
- Laser–Leistung in Abhängigkeit von der Bahngeschwindigkeit
  ...

**Auswertefunktion SYNFCT()**

Die Funktion hat die folgenden Parameter:

**SYNFCT**( Polynom–Nummer,
Echtzeitvariable–Ausgang,
Echtzeitvariable–Eingang)

Die Definition eines Polynom finden Sie in 2.4.4.

**Wirkungsweise SYNFCT**

Das mit ’Polynom–Nummer’ bestimmte Polynom wird mit dem Wert der ’Echtzeitvariable–Eingang’ ausgewertet. Das Ergebnis wird dann nach oben und nach unten begrenzt und der ’Echtzeitvariable–Ausgang’ zugewiesen.

Beispiel:
FCTDEF(1,0,100,0,0.8,0,0) ; Polynom 1 Definition sei erfolgt
...
Synchronaktion:

ID=1 DO **SYNFCT**(1,$AA_VC[U1], $A_INA[2])
; der additive Korrekturwert der Achse U1 wird in jedem Interpolationstakt über Polynom 1 aus dem Analogeingangswert 2 berechnet

Als ’Echtzeitvariable–Ausgang’ können Variable gewählt werden, die:

- mit additiver Beeinflussung (z.B Vorschub)
- mit multiplikativer Beeinflussung (z.B. Override)
- als Positionsoffset
- direkt

in den Bearbeitungsvorgang eingehen.

**Additive Vorschub Beeinflussung**

Bei der additiven Beeinflussung wird der programmierte Wert (bei der AC–Regelung das F–Wort) **additiv** korrigiert. $F_{wirksam} = F_{programmiert} + F_{AC}$
Als ’Echtzeitvariable–Ausgang’ werden z.B. gesetzt:
$AC_VC additive Bahnvorschubkorrektur,
$AA_VC[Achse] additive axiale Vorschubkorrektur

### Beispiel additive Beeinflussung des Bahnvorschubes

Der programmierte Vorschub (gleich ob axial– oder bahnbezogen) soll **additiv** vom Strom (positiven) der X–Achse (z. B. Zustellmoment) geregelt werden. Der Arbeitspunkt wird auf 5 A festgelegt. Der Vorschub darf ± 100 mm/min verändert werden, wobei die Abweichung des axialen Stromes ± 1 A betragen darf.

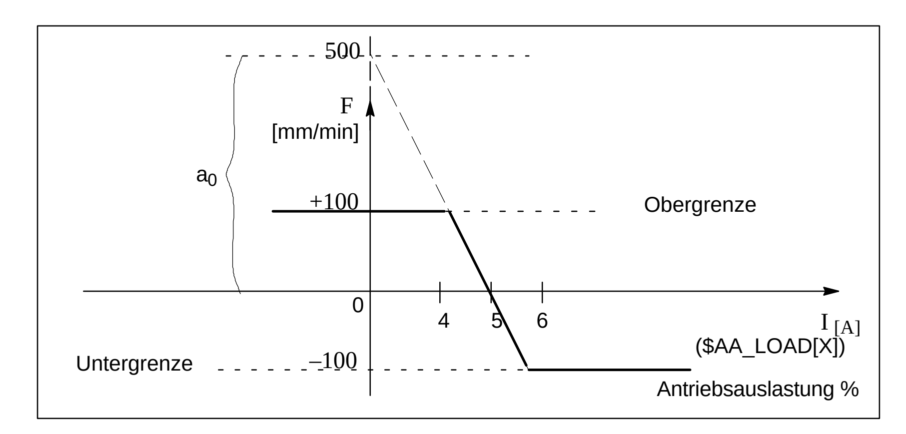



Bild 2-4 Beispiel additive Beeinflußung

Bestimmung der Koeffizienten s. auch 2.4.4:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -\frac{100\ \mathrm{mm}}{1\ \mathrm{min} \cdot \mathrm{A}}$

$a_1 = -100 \Rightarrow$ Regelkonstante

$a_0 = -(-100) \cdot 5 = 500$

$a_2 = 0$ (kein quadratisches Glied)

$a_3 = 0$ (kein kubisches Glied)

Obergrenze = 100

Untergrenze = –100

Damit ist das zu definierende Polynom (Nr. 1):

FCTDEF(1, –100, 100, 500, –100, 0, 0)

Mit dieser Funktion ist das Beispiel Bild 2-4 vollständig beschrieben.

Mit folgender Synchronaktion wird die *AC–Regelung* eingeschaltet:
ID = 1 DO **SYNFCT**(1, $AC_VC[x], $AA_LOAD[x])
; der additive Korrekturwert für den Vorschub der Achse x wird in jedem Interpolationstakt über Polynom 1 aus dem prozentualen Auslastungswert des Antriebes berechnet

### Multiplikative Beeinflussung

Bei der multiplikativen Beeinflussung wird das F–Wort mit einem Faktor (bei der AC–Regelung der Override) **mulipliziert**. $F_{wirksam} = F_{programmiert} \cdot Faktor_{AC}$

Als Echtzeitvariable–Ausgang wird die *multiplikativ* auf die Bearbeitung wirkende Variable $AC_OVR verwendet.

**Beispiel Multiplikative Beeinflussung**

Der programmierte Vorschub (gleich ob axial– oder bahnbezogen) soll **multiplikativ** in Abhängigkeit von der Antriebsauslastung beeinflußt werden. Der Arbeitspunkt wird dabei auf 100 % bei 30 %–iger Auslastung des Antriebs festgelegt. Bei 80 %–iger Auslastung soll die Achse (n) stehen. Eine Überhöhung der Geschwindigkeit wird mit +20 % der programmierten Geschwindigkeit zugelassen.

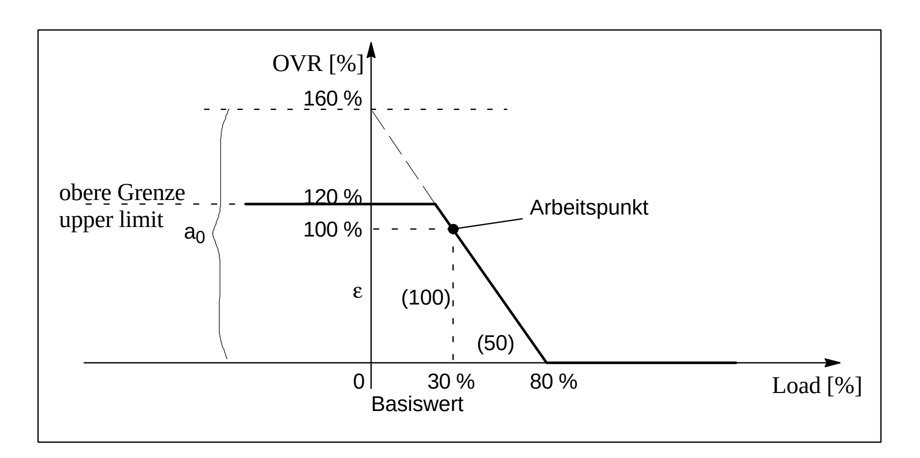



Bild 2-5 Beispiel multiplikative Beeinflußung

Bestimmung der Koeffizienten s. auch 2.4.4:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -\frac{100\ \%}{(80 - 30)\ \%} = -2$

$a_0 = 100 + (2 \cdot 30) = 160$

$a_2 = 0$ (kein quadratisches Glied)

$a_3 = 0$ (kein kubisches Glied)

Obergrenze = 120

Untergrenze= 0

Damit kann das Polynom (Nr. 2) definiert werden:

FCTDEF(2, 0, 120, 160, –2, 0, 0)

Mit dieser Funktion ist das Beispiel Bild 2-5 vollständig beschrieben.

Die dazugehörige Synchronaktion kann wie folgt lauten:
ID = 1 DO **SYNFCT**(2, $AC_OVR, $AA_LOAD[x])
; der Bahnoverride wird in jedem Interpolationstakt über Polynom 2 aus der prozentualen Auslastung des Antriebes für die x–Achse berechnet

**Positionsoffset mit Begrenzung**

Die Systemvariable $AA_OFF steuert eine achsspezifische Überlagerung, die sofort wirkt (Basis–Koordinatensystem). Die Art der Überlagerung wird durch:

MD 36750: $MA_AA_OFF_MODE

festgelegt.

0: proportionale Bewertung
1: integrale Bewertung

Ab SW–Stand 4 ist es möglich, den absolut zu korrigierenden Wert (Echtzeitvariable–Ausgang) auf den im Settingdatum

SD 43350 : $SA_AA_OFF_LIMIT

hinterlegten Wert zubegrenzen.
Ob die Begrenzung erreicht wird, kann durch Auswertung der achsspezifischen Systemvariablen

$AA_OFF_LIMIT[Achse]

in einer (weiteren) Synchronaktion abgefragt werden.

Wert –1: Limit des Korrekturwertes wurde in negativer Richtung erreicht.
Wert 1: Limit des Korrekturwertes wurde in positver Richtung erreicht.
Wert 0: Der Korrekturwert ist nicht im Grenzbereich.

Anwendung:
Die Funktion SYNFCT in Verbindung mit der Systemvariablen $AA_OFF kann für eine Abstandsregelung in der Laserbearbeitung benutzt werden. S. u.

**Beispiel**

Aufgabe:
**Abstandsregelung** als Funktion eines Sensorsignales bei Laser–Bearbeitung
Der Korrekturwert wird in negativer Z–Richtung begrenzt, damit sich der Laserkopf nicht in bereits gefertigten Blechausschnitten verhakt. Bei erreichtem Grenzwert können Anwenderreaktionen wie Achsen Stoppen (mittels Override 0, s.2.4.11) oder Alarm setzen s. 2.4.20 ausgelöst werden.
Randbedingungen:
*Integrierende* Bewertung der Eingangsgröße vom Sensor $A_INA[3].
Die Korrektur wirkt im Basiskoordinatensystem, d.h. vor der kinematischen Transformation. Ein evtl. programmierter Frame (TOFRAME) wirkt nicht, d.h. die Funktion kann nicht für eine 3D–Abstandsregelung in Orientierungsrichtung verwendet werden. Eine Abstandsregelung mit hohen Dynamikanforderungen oder eine 3D–Abstandsregelung kann mit der Funktrion ”Abstandsregelung” realisiert werden. Siehe

**Literatur:** /FB/, TE1, ”Abstandsregelung”

**Literatur:** /PG/, ”Programmieranleitung Grundlagen”

Die Abhängigkeit zwischen Eingangsgröße und Ausgangsgröße sei gegeben durch den im folgenden Bild dargestellten Zusammenhang.

**Weitere Beispiele**

In 6.3.1 finden Sie ein Beispiel mit dynamischer Anpassung einer Grenze des Polynoms bei der AC–Regelung (Abstandsregelung). In 6.3.2 finden Sie ein Beispiel für AC–Regelung des Bahnvorschubes.

**Abstandsregelung**

Der Abstandswert wird über das MD 36750: AA_OFF_MODE[V]=1 integrierend verrechnet. Es wirkt im Basiskoordinatensystem, d. h. vor der Transformation. Damit kann man es als Abstandsregelung in Orientierungsrichtung verwenden (nach Frameanwahl mit TOFRAME).

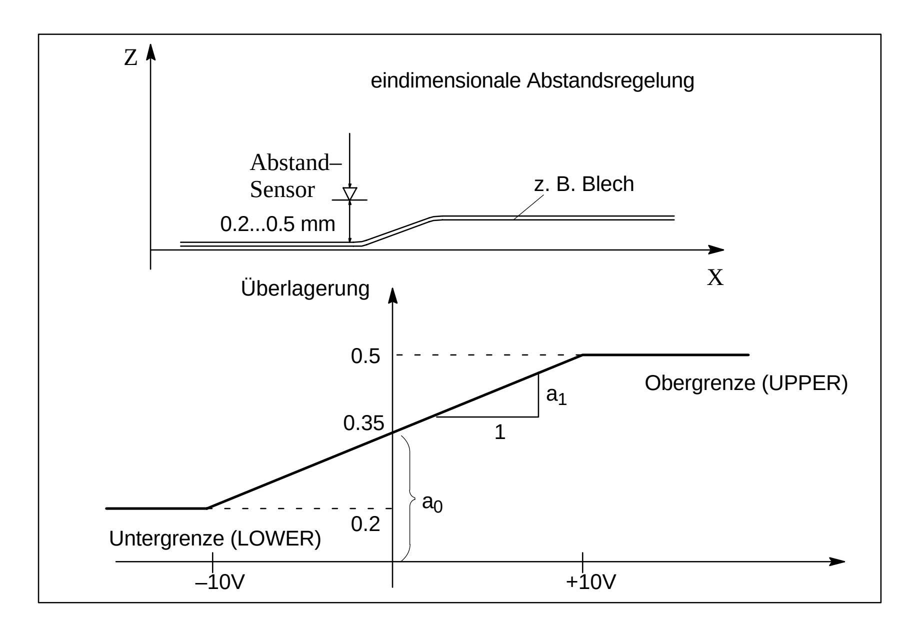



Bild 2-6 Abstandsregelung

```
%_N_AON_SPF
PROC AON                             ; Unterprogramm für Abstandsregelung ein
FCTDEF(1, 0.2,  0.5, 0.35, 1.5 EX–5) ; Polynomdefinition: Die Korrektur erfolgt
                                     ;im Bereich 0.2 bis 0.5

ID=1 DO SYNFCT(1,$AA_OFF[Z], $A_INA[3])       ; Abstandsregelung aktiv
ID = 2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO $AA_OVR[X] = 0
                                     ; Bei Überschreitung des Grenzbereiches X
                                     ; sperren.
RET
ENDPROC
```

```
%_N_AOFF_SPF
PROC AOFF                            ; Unterprogramm für Abstandsregelung aus
CANCEL(1)                            ; Synchronaktion Abstandsregl. löschen
CANCEL(2)                            ; Grenzbereichsprüfung löschen
RET
ENDPROC
```

```
%_N_MAIN_MPF; Hauptprogramm
                                     ; MD 36750 sei vor Power On auf 1 für
                                     ; integrierende Bearbeitung gesetzt.
$SA_AA_OFF_LIMIT[Z]= 1               ; Grenzwert für die Korrektur
AON                                  ; Abstandsregelung ein
...
G1 X100 F1000
AOFF                                 ; Abstandsregelung aus
M30
```

### 2.4.6 Überlagerte Bewegungen $AA_OFF einstellbar (ab SW 6)

**Überlagerte Bewegungen bis SW 5.3**

Unabhängig vom aktuellen Werkzeug und der Bearbeitungsebene ist über die Systemvariable $AA_OFF eine überlagerte Bewegung für jede Achse des Kanals möglich. Die Verschiebung wird sofort herausgefahren, unabhängig davon, ob die Achse programmiert ist oder nicht. Damit kann eine Abstandsregelung realisiert werden.

Mit dem achsialen MD 36750: AA_OFF_MODE wird die Art der Verrechnung wie folgt festgelegt:
Bit0 = 0: proportionale Verrechnung (absoluter Wert)
Bit0 = 1: integrierende Verrechnung (inkrementeller Wert)

$AC_VACTB und $AC_VACTW als Eingangsvariable für Synchronaktionen und die Ausgabe werden über Optionsbit verriegelt (”Vorschubabhängige Analogwertsteuerung” ⇒ Laserleistungssteuerung)!

$AA_OFF, Positionsoffset als Ausgangsvariable für Synchronaktionen für die Abstandsregelung wird über Optionsbit verriegelt!

Geschwindigkeitsbegrenzung mit MD 32070: CORR_VELO.

**Verhalten von $AA_OFF ab SW 6**

**Nach RESET** kann der Positionsoffset weiterhin erhalten bleiben

Bisher wurde bei RESET der Positionsoffset von $AA_OFF abgewählt. Da dieses Verhalten bei statischen Synchronaktionen IDS = <Nummer> DO $AA_OFF = <Wert> zu einer sofortigen erneuten überlagerten Bewegung mit der Interpolation eines Positionsoffset führt, kann über das Maschinendatum MD 36750: AA_OFF_MODE das Verhalten von RESET eingesetellt werden.

Bit1 = 0: $AA_OFF wird bei RESET abgewählt
Bit1 = 1: $AA_OFF bleibt über RESET hinaus erhalten

**In der Betriebsart JOG** kann eine überlagerte Bewegung stattfinden

Auch in der Betriebsart JOG kann bei einer Änderung von $AA_OFF eine Interpolation des Positionsoffset als überlagerte Bewegung über das Maschinendatum MD 36750: AA_OFF_MODE eingesetellt werden.

Bit2 = 0: keine überlagerte Bewegung aufgrund von $AA_OFF
Bit2 = 1: eine überlagerte Bewegung aufgrund von $AA_OFF

Wird ein Positionsoffset aufgrund von $AA_OFF interpoliert, so kann eine Betriebsartenumschaltung nach JOG erst erfolgen, wenn die Interpolation des Positionsoffsets beendet ist. Anderenfalls wird der Alarm 16907 gemeldet.

**Aktivierung/ Deaktivierung**

Die programmierten Bedingungen der aktuellen Bewegungssynchronaktionen werden im IPO–Takt erfaßt, bis die Bedingungen erfüllt sind oder das Ende des nachfolgenden Satzes mit Maschinenfunktion errreicht ist.

Ab Software–Stand 3.2 erfolgt mit Einführung einer für Synchronaktionen zugelassenen $$–Hauptvariablen ein Vergleich der Synchronisations– bedingungen im IPO–Takt im Hauptlauf.

**Randbedingungen**

- **Interruptroutinen/asynchrone Unterprogramme**
Bei Aktivierung einer Interruptroutine bleiben modale Bewegungssynchronaktionen erhalten und sind auch im asynchronen Unterprogramm wirksam. Erfolgt der Unterprogrammrücksprung nicht mit REPOS, so wirken im Hauptprogramm die im asynchronen Unterprogramm geänderten modalen Synchronaktionen weiter.

- **REPOS**
Im Restsatz gelten die Synchronaktionen wie im Unterbrechungssatz. Änderungen an den modalen Synchronaktionen im asynchronen Unterprogramm sind im unterbrochenen Programm nicht wirksam. Die mit FCTDEF programmierten Polynomkoeffizienten werden von ASUP und REPOS nicht beeinflusst.

  Im asynchronen Unterprogramm wirken die Koeffizienten aus dem aufrufenden Programm. Im aufrufenden Programm wirken die Koeffizienten aus dem asynchronen Unterprogramm weiter.

- **Programmende**
Die mit FCTDEF programmierten Polynomkoeffizienten wirken über Programmende hinweg.

- **Satzsuchlauf**
Bei Satzsuchlauf mit Berechnung werden diese Polynomkoeffizienten aufgesammelt, d.h. in die Settingdaten geschrieben.

**CORROF ab SW 6**

- Der Teileprogrammbefehl CORROF mit DROF wird beim Satzsuchlauf mit aufgesammelt und in einem Aktionssatz ausgegeben. Dabei werden in den letzten vom Suchlauf behandelten Satz mit CORROF oder DROF alle abgwählten DRF–Verschiebungen aus Gründen der Kompatibilität aufgesammelt.

  Ein CORROF mit AA_OFF wird beim Satzsuchlauf nicht aufgesammelt und geht verloren. Will ein Anwender diesen Suchlauf weiterhin nutzen, so ist dies mit den Satzsuchlauf via Programmtest ”SERUPRO” möglich. Weitere Details zu diesen Satzsuchlaufs sind beschrieben in:

**Literatur:** /FB1/, K1 ”BAG, Kanal, Programmmbetrieb”, Programmtest

- **DRF–Verschiebungen achsspezifischmit CORROF abwählen**
Mit CORROF sind DRF–Verschiebungen für die einzelne Achsen nur vom Teileprogramm aus möglich.

- **Abwahl des Positionsoffsets bei aktiven Synchronaktionen**
Ist bei der Abwahl des Positionsoffsets über den Teileprogrammbefehl CO-ROFF(Achse,”AA_OFF”) eine Synchronaktion aktiv, so wird der Alarm 21660 gemeldet. Gleichzeitig wird $AA_OFF abgewählt und nicht wieder gesetzt. Wird die Synchronaktion später im Satz nach CORROF aktiv, so bleibt $AA_OFF gesetzt und es wird ein Positionsoffset interpoliert.

**Literatur:** /PG/, ”Programmieranleitung Grundlagen”

---

**Hinweis**

Abhängig davon, in welchem Koordinatensystem (BKS oder WKS) eine Echtzeitvariable definiert ist, werden Frames eingerechnet oder nicht.

Entfernungen werden immer im eingestellten Grundsystem (metrisch oder inch) berechnet. Eine Umschaltung mit G70 oder G71 hat keine Auswirkung.

DRF–Verschiebungen, externe Nullpunktverschiebungen usw. werden nur bei Echtzeitvariablen berücksichtigt, die im MKS definiert sind.

---

## 2.4.7 Online–Werkzeugkorrektur FTOC

**Online–Werkzeug–korrektur**

Bei der Technologie Schleifen kann die Bearbeitung des Werkstückes und das Abrichten der Schleifscheibe im gleichen Kanal oder in unterschiedlichen Kanälen (Bearbeitungs– und Abrichtkanal) durchgeführt werden.

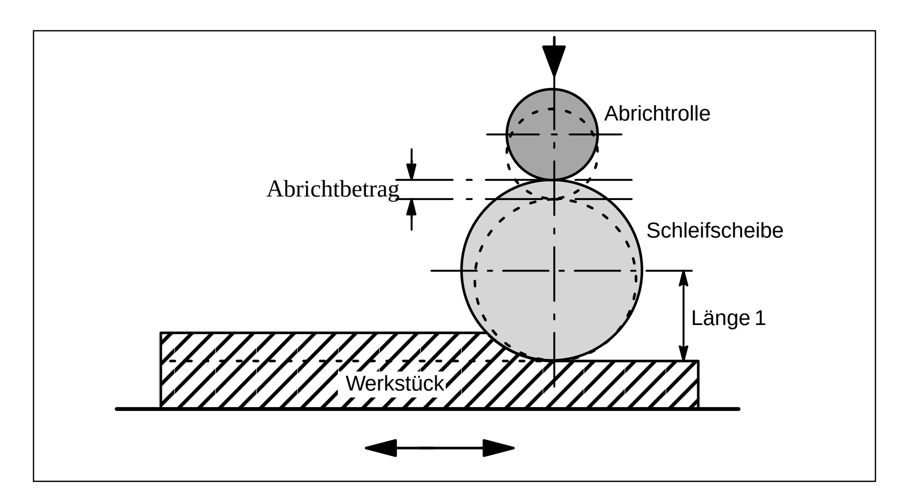



Bild 2-7 Abrichten während der Bearbeitung mit einer Abrichtrolle

**Literatur:** /FB/, W4 ”Schleifen”

**Randbedingung**

Die Synchronaktion FTOC steht ab Software–Stand 3.2 zu Verfügung.

Die Online–Korrektur ermöglicht eine überlagerte Bewegung für eine Geometrieachse nach einem mit FCTDEF programmierten Polynom (s. 2.4.4 ) in Abhängigkeit von einem Bezugswert, der z. B. der Istwert einer Achse sein kann.

**Programmierung FTOC**

Die Online–Korrektur wird wie folgt angegeben:

**FTOC**( Polynom–Nr,
Real–Hauptvariable_lesen, ;Bezugswert
Länge 1_2_3,
Kanalnummer,
Spindelnummer)

**Parameter**

| | |
|---|---|
| Polynom–Nr: | Nummer der zuvor mit FCTDEF parametrierten Funktion. |
| Real–Hauptvariable_lesen: | Es sind alle bei 2.3.8 aufgeführten Hauptvariablen vom Typ REAL zulässig. |
| Länge 1_2_3: | Verschleißparameter, in dem der Korrekturwert **addiert** wird. |

| | |
|---|---|
| Kanalnummer: | Zielkanal, in dem die Korrektur wirken soll. Damit ist zeitgleiches Abrichten aus einem parallelen Kanal möglich. Entfällt die Kanalnummer, so wirkt die Korrektur im aktiven Kanal. Im Zielkanal der Korrektur muß die Online–Korrektur mit FTOCON eingeschaltet sein. |
| Spindelnummer: | Die Spindelnummer wird programmiert, wenn eine nicht aktive Schleifscheibe abgerichtet werden soll. Voraussetzung ist, daß ”konstante Scheibenumfangsgeschwindigkeit” oder ”Werkzeugüberwachung” aktiv ist. Wird keine Spindelnummer programmiert, so wird das aktive Werkzeug korrigiert. |

Beispiel

Länge einer aktiven Schleifscheibe korrigieren

```
%_N_ABRICHT_MPF

FCTDEF(1,–1000,1000,–$AA_IW[V],1)            ;Definition der Funktion
ID=1 DO FTOC(1,$AA_IW[V],3,1)   ; Online– Werkzeugkorrektur anwählen:
                                ; abgeleitet von der Bewegung der V–Achse
                                ; wird in Kanal 1 die Länge 3 der aktiven
                                ; Schleifscheibe korrigiert.
WAITM (1,1,2)                   ; Synchronisation mit Bearbeitungskanal
G1 V–0.05 F0.01, G91
G1 V –....
...
CANCEL(1)                       ; Online–Korrektur abwählen
...
```

---

**Hinweis**

Es wird kein Häufigkeitskennwort und keine Bedingung in der Synchronaktion angegeben, damit wird die Aktion FTOC in jedem Interpolationstakt ohne weitere Abhängigkeiten als von $AA_IW[V] wirksam.

---

### 2.4.8 RDISABLE

**Programmierte Einlesesperre RDISABLE**

Der RDISABLE–Befehl im Aktionsteil bewirkt, daß die weitere Satzbearbeitung angehalten wird, wenn die zugehörige Bedingung erfüllt ist. Es werden nur noch die programmierten Bewegungssynchronaktionen bearbeitet. Wenn die Bedingung für die RDISABLE–Anweisung nicht mehr erfüllt ist, wird die Einlesesperre aufgehoben.

Am Ende des Satzes mit RDISABLE wird Genauhalt ausgelöst, unabhängig davon, ob die Einlesesperre wirksam wird oder nicht.
**Anwendung:** Damit kann z. B. abhängig von externen Eingängen das Programm im Interpolationstakt gestartet werden.

Beispiel RDISABLE

Programmierte Einlesesperre

```
WHENEVER $A_INA[2]<7000 DO RDISABLE
...
N10 G01 X10      ;Am Ende von N10 wirkt RDISABLE, wenn während seiner
                   Bearbeitung die Bedingung erfüllt ist
N20 Y20
```

Wenn die Spannung 7V am Eingang 2 unterschreitet, wird die Programmfortsetzung angehalten (Annahme: Wert 1000 entspricht 1V).
Anwendung dieser Lösung z.B: Einlesesperre bis Hindernis aus dem Weg geräumt ist.

### 2.4.9 STOPREOF

**Beendigung des Vorlaufstop STOPREOF**

Eine Bewegungssynchonaktion mit einem STOPREOF–Befehl hebt den bestehenden Vorlaufstop auf, wenn die Bedingung erfüllt ist.
STOPREOF darf nur mit dem Schlüsselwort 'WHEN' und satzweise wirksam programmiert werden.
**Anwendung:** Schnelle Programmverzweigung am Satzende.

Beispiel STOPREOF

Programmverzweigungen

```
WHEN $AC_DTEB<5 DO STOPREOF
G01 X100
IF $A_INA[7]>5000 GOTOF Label 1
```

Wenn die Entfernung zum Satzende 5 mm unterschreitet, beende den Vorlaufstop. Wenn die Spannung 5V am Eingang 7 überschreitet, springe vorwärts bis zum Label 1 (Annahme: Wert 1000 entspricht 1V).

### 2.4.10 DELDTG

**Restweglöschen**

Mit Synchronaktionen kann in Abhängigkeit von einer Bedingung Restweglöschen für die **Bahn** und für angegebene **Achsen** ausgelöst werden.

- Schnelles vorbereitetes Restweglöschen

**Für die Bahn schnelles, vorbereitetes RWL**

Schnelles / vorbereitetes Restweglöschen wird eingesetzt bei zeitkritischen Anwendungen:

– wenn die Zeit zwischen Restweglöschen und Start des Folgesatzes sehr kurz sein soll

– wenn Restweglöschen mit sehr hoher Wahrscheinlichkeit ausgelöst wird

**DELDTG**

Die Programmierung erfolgt mit der Synchronaktion **DELDTG.**
Nach Ausführung von Restweglöschen steht in der Systemvariable $AC_DELT der Bahnrestweg. Damit wird Bahnsteuerbetrieb am Ende des Satzes mit schnellem Restweglöschen unterbrochen.

Einschränkungen:
Restweglöschen für die Bahn kann nur als satzweise wirksame Synchronaktion programmiert werden.
Bei aktiver Werkzeugradiuskorrektur kann schnelles Restweglöschen nicht verwendet werden.

Befehle: MOVE=1:Geht bei Teilungsachsen mit und ohne Hirthverzahnung MOV=0: Funktioniert bei beiden gleich, es wird die nächste Position angefahren. Befehl:DELDTG:Bei Teilungsachsen ohne Hirthverzahnung: Achse steht sofort. Bei Teilungsachsen mit Hirthverzahnung: Achse fährt nächste Position an.

Beispiel
DELDTG

```
... DO DELDTG
N100 G01 X100 Y100 F1000
N110 G01 X...
IF $AC_DELT > 50
...
```

**Für Achsen schnelles, vorbereitetes RWL**

Schnelles, vorbereitetes Restweglöschen für Achsen kann nur satzweise erfolgen.

Anwendung:
Das Stoppen einer Positionierbewegung, die im Teileprogramm programmiert wurde, erfolgt mit axialem Restweglöschen. Mit einem Befehl können mehrere Achsen gleichzeitig gestoppt werden.

... DO **DELDTG(Achse1, Achse2, ...)**

Beispiele
DELDTG(Achse)

```
WHEN $A_INA[2]>8000 DO DELDTG(X1)
                              ; wenn an Eingang 2 die Spannung
                              ; von 8 V überschritten wird, Restweg
                              ; löschen für Achse X1
POS[X1] = 100                 ; nächste Position
R10 = $AA_DELT[X 1]           ; Übernehmen axialen Restweg in R10
```

Nach erfolgtem Restweglöschen enthält die Variable $AA_DELT[Achse] den achsialen Restweg.

(Annahme: Wert 1000 entspricht 1V).

## 2.4.11 Sperren einer programmierten Achsbewegung

**Aufgabe**

Die Achse ist innerhalb eines Bearbeitungsprogramms programmiert und soll in speziellen Fällen nicht am Satzbeginn starten.

**Lösungsmethode**

Per Synchronaktion wird der Override bis zum Startzeitpunkt auf 0 gehalten.

Beispiel:

```
WHENEVER  $A_IN[1]==0 DO $AA_OVR[W]=0
G01 X10 Y25 F750  POS[W]=1500  FA[W]=1000
                    ; Die Positionierachse wird asynchron zur
                    ; Bahnbearbeitung gestartet;
                    ; die Freigabe erfolgt über einen digitialen Eingang
```

---

**Hinweis**

Das Sperren einer Achsbewegung ist auch für PLC–Achsen möglich (z.B. Magazinachse).

---

## 2.4.12 Starten von Kommandoachsen

**Einführung**

Achsen können auch vollkommen asynchron zum Teileprogramm aus Synchronaktionen positioniert, gestartet gestoppt werden. Diese Art der Programmierung empfiehlt sich für zyklische Abläufe oder Abläufe, die sehr stark ereignisgesteuert sind. Achsen, die aus Synchronaktionen gestartet werden, heissen **Kommandoachsen.**

**Kontrolle vom PLC**

**Autarke Einzelachsvorgänge (ab SW 6.3)**

Eine vom Hauptlauf interpolierende Kommandoachse (gestartet über statische Synchronaktionen) reagiert unabhängig vom NC–Programm bei NC–STOP, Alarmbehandlung, Programmende, Programmbeeinflussungen und RESET, wenn die Kontrolle der Kommandoachse vom PLC übernommen wurde.

Die Kontrolle über die Kommandoachse erfolgt über die axiale VDI–Nahtstelle (PLC→NCK) mit den NST ”PLC kontrolliert Achse” (DB31, ... DBX28.7) == 1

Weitere Informationen über den genauen Ablauf der verschiedenen Schritte die Kontrolle der Kommandoachse an die PLC zu übertragen siehe bitte:
**Literatur:** /FB/, P2, ”Positionierachsen”

**Randbedingung**

Eine Achse kann nicht gleichzeitig aus dem Teileprogramm und aus Synchronaktionen bewegt werden. Zeitlich nacheinander ist dies möglich.
Wartezeiten können auftreten, wenn eine Achse nach einer Bewegung aus Synchronaktionen wieder im Teileprogramm programmiert wird.

**Hinweis**

Das MD 30450: IS_CONCURRENT_POS_AX gibt darüber Auskunft, ob die Achse hauptsächlich als Kommandoachse oder für Programmierung durch das Teileprogramm vorgesehen ist:

0: keine konkurrierende Achse
1: konkurrierende Achse (Kommando–Achse)

**Beispiel 1**

...
ID=1 EVERY $A_IN[1]==1 DO **POS[X]=100**
...

**Beispiel 2**

Eine Achsbewegung kann als Technologiezyklus ausgelöst werden.(S. 2.5)
Hauptprogramm:

...
ID=2 EVERY $A_IN[1]==1 DO **ACHSE_X**
...

Achsprogramm:

ACHSE_X:
M100
POS[X]=100
M17

**Programmierung**

Positionierachsbewegungen werden in der Synchronaktion wie aus dem Teileprogramm programmiert:

ID = 1 EVERY $AA_IM[B] > 75 DO **POS[U]=100**

Die programmierte Position wird inch oder metrisch bewertet entsprechend der im aktuell bearbeiteten Teileprogrammsatz gültigen Einstellung G70 oder G71.

G70/G71 sowie G700/G710 können ab SW 5 auch in den Synchronaktionen direkt programmiert werden.

Damit kann die Inch–, Metrisch–Bewertung einer Kommandoachs–Bewegung unabhängig von der Programmierung im Teileprogramm festgelegt werden.

ID = 1 WHENEVER $A_OUT[1] ==1 DO **G710** POS[X]=10

ID = 2 EVERY **G710** $AA_IM[Z] >100 DO **G700** POS[Z2]=10

**Hinweis**

Es sind nur **G70, G71, G700, G710** in Synchronaktionen programmierbar! Vergl. Kap. 2.1.

G–Funktionen, die im Synchronaktionssatz programmiert werden, wirken nur auf die Synchronaktion oder im Technologiezyklus. Sie haben keine Auswirkung auf die folgenden Sätze im Teileprogramm.

**Literatur:** /PG/ Kap. 3 ”Wegangaben”

**Endposition absolut / inkrementell**

Die Endposition kann absolut oder inkrementell programmiert werden. Je nachdem, ob im gerade aktiven Satz des Hauptprogramms G90 oder G91 aktiv ist, wird die Position absolut oder inkrementell verfahren.
Es kann auch bei der Programmierung explizit bestimmt werden, ob der Wert absolut oder inkrementell programmiert wird:

**IC**: inkrementell
**AC**: absolut
**DC**: direkt, d.h. Rundachse auf kürzestem Weg positionieren
**ACN**: Modulo–Rundachse absolut positionieren in negativer Bewegungsrichtung
**ACP**: Modulo–Rundachse absolut programmieren in positiver Bewegungsrichtung
**CAC**: Achse auf codierte Position verfahren absolut
**CIC**: Achse auf codierte Position verfahren inkrementell
**CDC**: Rundachse auf kürzestem Weg auf codierte Position verfahren
**CACN**: Modulo–Rundachse in negativer Richtung auf codierte Position verfahren
**CACP**: Modulo–Rundachse in positiver Richtung auf codierte Position verfahren

Codierte Positionen sind in Maschinendaten hinterlegte Werte.

**Beispiel 1**
**fester Wert**

ID = 1 EVERY $AA_IM[B] > 75  DO **POS[U]=IC(10)**

; Wenn Ereignis eintritt, U–Achse um 10 weiterpositionieren

**Beispiel 2**
**aktueller Wert**

Der zu verfahrende Weg wird in Echtzeit aus einer Echtzeitvariablen gebildet:

ID = 1 EVERY $AA_IM[B] > 75  DO **POS[U]=$AA_MW[V]–$AA_IM[W] + 13.5**

**Achsiale Frames**

Das Verhalten von Synchronaktionen und achsialen Frames erläutern die folgenden Abschnitte:

**Wirkung**

Bei Positionierbewegungen aus Synchronaktionen wirken die achsialen Verschiebungen, Skalierungen und Spiegelungen des programmierbaren und einstellbaren Frames (G54 usw.) sowie Werkzeuglängenkorrekturen.

Es wirkt jeweils der im aktiven Satz wirksame Frame. Ist im aktiven Satz eine Drehung aktiv, so wird das Auslösen einer Positionierbewegung aus der Synchronaktion mit Alarm abgelehnt.

**Beispiel:**

```
TRANS X20
IDS= 1 EVERY $A_IN==1 DO POS[X]=40
G1 Y100          ;Wenn der Eingang gesetzt wird, wird X auf 60 positioniert
...
TRANS X–10
G1 Y10           ; Wenn der Eingang gesetzt wird, wird X auf 30 positioniert
```

**Unterdrückung**

Die Wirksamkeit von Frames und Werkzeuglängen kann unterdrückt werden mit

MD 32074: FRAME_OR_CORRPOS_NOTALLOWED

**axiale Frames unterdücken**

Auf eine Kommandoachse sind axiale Frames nicht wirksam die inkrementell auf Teilungspositionen verfahren. Deshalb wird im

MD 32074: FRAME_OR_CORRPOS_NOTALLOWED[AX4] das Bit 9 =1 gesetzt

und die Kommandoachse mit JOG positioniert.

**Beispiel:**

```
RANS A=0,001
POS[A]=CAC(2)                    ; Achse fährt auf Position 180.001 Grad

; Für die Kommandoachse ist das axiale Frame nicht wirksam
; MD 32074: FRAME_OR_CORRPOS_NOTALLOWED[AX4] = ’H0020’

WHEN TRUE DO POS[A]=CIC(–1) ; Achse fährt auf Position 180.000 Grad.
```

---

**Hinweis**

Wird einen Kommandoachse inkremetell auf Teilungspositionen verfahren, dann sind axiale Frames generell für diese Kommandoachse **nicht** wirksam.

---

## 2.4.13 Axialer Vorschub aus Synchronaktionen

**Vorschübe**

Zusätzlich zur Endposition kann ein achsialer Vorschub programmiert werden:

ID = 1 EVERY $AA_IM[B] > 75 DO POS[U]=100 **FA[U]=990**

Der achsiale Vorschub für Kommandoachsen ist modal wirksam. Er wird unter der Adresse FA programmiert. Der Standardwert wird über das axiale
MD 32060: POS_AX_VELO
vorgegeben.

Der Vorschubwert wird entweder fest vorgegeben oder in Echtzeit aus Echtzeitvariablen gebildet:

**Beispiel für berechneten Vorschub**

ID = 1 EVERY $AA_IM[B] > 75 DO POS[U]=100 **FA[U]=$AA_VACTM[W]+100**

Der Vorschubwert wird entweder als Linear– oder Umdrehungsvorschub programmiert:
Den Vorschubtyp bestimmt das Settingdatum:
SD 43300: $SA_ASSIGN_FEED_PER_REV_SOURCE.
Das Settingdatum kann per Bedienung oder von PLC, sowie aus dem Teileprogramm verändert werden. Synchron zumTeileprogrammkontext kann der Vorschubtyp über die Sprachbefehle FPRAON, FPRAOF umgeschaltet werden. Siehe dazu:

**Literatur:** /FB/, V1 ”Vorschübe”

---

**Hinweis**

Der achsiale Vorschub aus Bewegungssynchronaktionen wird nicht als Hilfsfunktion an die PLC ausgegeben. Parallele achsiale Technologiezyklen würden sich sonst gegenseitig blockieren.

---

### 2.4.14 Achsen aus Synchronaktionen starten / stoppen

**Starten/Stoppen**

Kommandoachsen können auch ohne Angabe einer Endposition aus Synchronaktionen gestartet werden. Die Achse wird dann solange in die programmierte Richtung verfahren, bis durch einen erneuten Bewegungs– oder Positionier–Befehl eine andere Bewegung vorgegeben wird, oder die Achse durch einen Stoppbefehl angehalten wird. Damit kann z.B. eine endlos drehende Rundachse programmiert werden.

Die Programmierung erfolgt analog zur Programmierung von Positionierbewegungen.

**MOV[Achse]=Wert**

Datentyp des Wertes ist INT.
Das Vorzeichen des Wertes bestimmt die Richtung der Bewegung:
> 0: Achsbewegung in positive Richtung
<0: Achsbewegung in negative Richtung
==0: Achsbewegung stoppen

Wird eine Teilungsachse aus der Bewegung mit MOV[Achse]=0 gestoppt, so wird wie im konventionellen JOG–Betrieb die nächste Teilungsposition angefahren.

Der **Vorschub** für die Bewegung kann programmiert werden mit **FA[Achse]=Wert**. (S. oben). Ist kein achsialer Vorschub programmiert, so ergibt sich der Wert aus einer evtl. bereits aus Synchronaktionen aktivierten Achs–Bewegung oder aus der über das axiale MD 32060: POS_AX_VELO eingestellten Achsgeschwindigkeit.

**Beispiel**

```
... DO MOV[u]=0          ;Achsbewegung stoppen, wenn Bedingung erfüllt ist
```

### 2.4.15 Spindelbewegungen aus Synchronaktionen

**Allgemeines**

Analog zu Positionierachsen können auch **Spindeln** aus Synchronaktionen gestartet, positioniert, gestoppt werden. Der Start von Spindelbewegungen zu definierten Zeitpunkten kann erreicht werden durch Blockieren einer im Teileprogramm programmierten Bewegung oder durch Steuerung der Achsbewegung aus Synchronaktionen.

**Starten/Stoppen**

Die Benutzung dieser Funktionen empfiehlt sich für zyklische Abläufe oder Abläufe, die sehr stark ereignisgesteuert sind.

**Stoppen bis Ereignis eintrifft**

Anwendungsfall:
Die Spindel ist innerhalb eines Bearbeitungsprogramms programmiert und soll in speziellen Fällen nicht am Satzbeginn starten. Per Synchronaktion wird der Override bis zu Startzeitpunkt auf 0 gehalten.
Beispiel:

```
ID=1 WHENEVER  $A_IN[1]==0 DO $AA_OVR[S1]=0
G01 X100 F1000 M3   S1=1000
                    ; Die Spindel wird asynchron zur Bahnbearbeitung gestartet;
                    ; der Start erfolgt über einen digitialen Eingang
```

**Hilfsfunktionen, Drehzahl, Position**

Die Programmierung erfolgt im Aktionsteil der Synchronaktion identisch mit der Programmierung im Teileprogramm.
Befehle: S= ..., M3, M4, M5, SPOS= ...
Beispiel:
ID = 1 EVERY $A_IN[1]==1 DO M3 S1000
ID = 2 EVERY $A_IN[2]==1 DO SPOS=270

Ohne numerische Erweiterung gelten die Befehle jeweils für die Masterspindel. Durch Angabe einer numerischen Erweiterung kann jede Spindel aktiviert werden:

ID = 1 EVERY $A_IN[1]==1 DO M1=3 S1=1000 SPOS[2]=90

Für die Programmierung der Art der Positionierung gelten dieselben Regeln wie für Positionierachsen (s.o.).

Werden durch parallel aktive Synchronaktionen für eine Achse/Spindel **konkurrierende Befehle** vorgegeben, so entscheidet die **zeitliche Reihenfolge** der Aktivierung.
Beispiel:

```
ID=1 EVERY $A_IN[1]==1 DO M3 S300 ; Drehrichtung und Drehzahl
ID = 2 EVERY $A_IN[2]==1 DO M4 S500; Drehrichtung und Drehzahl
ID=3 EVERY $A_IN[3]==1 DO S1000 ; Neue Drehzahlvorgabe
                                 ; für aktive Spindeldrehung
ID=4 EVERY ($A_IN[4]==1 ) AND ($A_IN[1]==0)  DO  SPOS=0
                                                  ; Spindel positionieren
```

**Vorschub**

Der Vorschub für Spindeln Positionieren kann aus der Synchronaktion programmiert werden mit:

**FA[Sn]**= ...

:

---

**Hinweis**

Für die Vorschubgeschwindigkeit aus Synchronaktionen steht nur ein modales Datum für Spindelbetrieb und Achsbetrieb zur Verfügung. Hierbei wird FA[S] bzw. FA[C] gleichermaßen versorgt.

---

**SW–Endschalter Arbeitsfeldbegrenzungen**

Für Achs–/Spindelbewegungen aus Synchronaktionen gelten auch die Beschränkungen durch SW–Endschalter und Arbeitsfeldbegrenzungen.

**Beachtung durch Bewegungen aus Synchronaktionen**

Die mit G25/G26 programmierten Arbeitsfeldbegrenzungen werden in Abhängigkeit vom Settingdatum:
SD 43400: $SA_WORKAREA_PLUS_ENABLE berücksichtigt.

Das Ein– und Ausschalten der Arbeitsfeldbegrenzung über G–Funktionen WALIMON / WALIMOF **im Teileprogramm** wirkt nicht auf Kommandoachsen.

---

**Achskoordinierung**

Wird aus Synchronaktionen ein Positionerbefehl (POS, MOV) gestartet, während die Achse bereits als Bahnachse oder als PLC–Achse belegt ist, so wird die Bearbeitung mit Alarm abgebrochen.

**Achsbewegung wechselweise durch TP und SA**

Typischerweise wird eine Achse entweder aus dem Teileprogramm (TP) im Bewegungssatz oder als Positionierachse aus der Synchronaktion (SA) bewegt. Soll dieselbe Achse jedoch wechselweise aus dem Teileprogramm als Bahnachse oder Positionierachse und aus Synchronaktionen verfahren werden, so erfolgt eine koordinierte Übergabe zwischen beiden Achsbewegungen.

**Beispiel**

**; X–Achse wahlweise aus Teileprogramm und Synchronaktionen fahren**

```
N10 G01 X100 Y200 F1000          ; X–Achse im Teileprogramm programmiert
...
N20  ID=1 WHEN $A_IN[1]==1  DO POS[X]=100  FA[X]=200
                                 ; Positionieren aus Synchronaktion starten,
                                 ; wenn digitialer Eingang ansteht
...
CANCEL(1)                        ; Synchronaktion abwählen
...

N100 G01 X100 Y200 F1000         ; X: Bahnachse
                                 ; Wartezeit vor Bewegung, wenn digitaler
                                 ; Eingang 1 war und somit X aus
                                 ; Synchronaktion positioniert wurde
```

---

**Fliegende Übergänge**

Zwischen Kommandoachsen und Spindeln sind Übergänge möglich.

**Ausgangssituation**

Da mehrere Synchronaktionen gleichzeitig aktiv sein können, ist es möglich, daß eine Achsbewegung gestartet wird, während die Achse bereits aktiv ist.

**Verhalten**

In diesem Fall wird die **zuletzt aktivierte** Bewegung wirksam. POS– und MOV–Bewegungen können sich gegenseitig ablösen.
Bei einer so erzwungenen Umkehr der Bewegungsrichtung wird die Achse zunächst abgebremst und dann in die entgegengesetzte Richtung positioniert.

Beispiele:

```
ID=1 EVERY $AC_TIMER[1] >= 5 DO POS[V]=100 FA[V]=560
ID=2 EVERY $AC_TIMER[1] >= 7 DO POS[V]=$AA_IM[V] + 2 FA[V]=790
```

; Aufgrund der Programmierung mit $AC_TIMER[1] ist die Synchronaktion mit ID=2 die zuletzt aktivierte, ihre Vorgaben werden wirksam und lösen die Vorgaben aus ID=1 ... ab.

Endposition und Vorschub einer Kommandoachse können somit während laufender Bewegung nachgestellt werden.

**Beispiel: Auslösung mit Signal**

```
ID=1    EVERY $A_IN[1]==1 DO POS[U]=$AA_IM[U]+$AA_IM[V]*.5
                   FA[U]=$AA_VACTM[U]+10
```

**Erlaubte Übergänge**

Die mit x gekennzeichneten Übergänge sind zulässig:

| im ↓ nach → | POS | MOV=1<br>MOV= – 1 | MOV=0 | SPOS | M3<br>M4 | M5 | LEADON | TRAIL–ON |
|---|---|---|---|---|---|---|---|---|
| **Achse steht** | | | | | | | | |
| Achsbetrieb | x | x | x | x | x | x | x | x |
| lagegeregelte Spindel | x | x | x | x | x | x | | |
| drehzahlgeregelte Spindel | | | | x | x | x | | |
| **Achse in Bewegung** | | | | | | | | |
| Achsbetrieb | x | x | x | | | | x | x |
| lagegeregelte Spindel | | | | | | | | |
| drehzahlgeregelte Spindel | | | | x | x | x | | |

Nicht gekennzeichnete Übergänge werden mit Alarm abgewiesen.

Beispiel: Erlaubter Übergang

```
N10 WHEN $AA_IM[Y] >= 5 DO MOV[Y]=–1          ; Bei Position +5 Achse in
                                              ; Negativer–Richtung
                                              ; starten
N20 WHEN TRUE DO POS[Y]=20 FA[Y]=500          ; Y–Achse starten, wenn
                                              ; Satz eingewechselt wird
```

**Fliegende Übergänge bei Achs–kopplungen**

Positionierachbewegungen und Bewegungen als Folge von Achskopplungen über Synchronaktionen können sich gegenseitig ablösen.

– siehe 2.4.17 und

**Literatur:** /M3/, Mitschleppen und Leitwertkopplungen

Die erlaubten Übergänge in Leitwertkopplung sind in der Tabelle oben mit LEADON gekennzeichnet, die Übergänge in das Mitschleppen mit TRAILON.

## 2.4.16 Istwertsetzen aus Synchronaktionen

**Anwendung**

Mit der Funktion PRESETON kann der Steuerungsnullpunkt im Maschinenkoordinatensystem neu definiert werden.

**Funktion**

Bei Preset findet keine Bewegung der Achse statt, es wird für die momentane Achsposition lediglich ein neuer Positionswert eingetragen.

**Programmierung**

In Synchronaktionen kann jeweils der Wert für **eine** Achse programmiert werden.
z.B:

**WHEN $AA_IM[a] >= 89.5 DO PRESETON(a, 10.5)**

mit PRESETON(Achse,Wert)
Achse: Achse, deren Steuerungsnullpunkt verändert werden soll
Wert: Wert, um den der Steuerungsnullpunkt verändert wird.

**Zulässige Anwendungen**

PRESETON aus Synchronaktionen ist möglich für:

- Modulo–Rundachsen, die aus dem Teileprogramm gestartet wurden
- alle Kommandoachsen, die aus der Synchronaktion gestartet wurden

**Einschränkung**

PRESETON ist nicht möglich für Achsen, die an der Transformation beteiligt sind.

**Beispiel**

In 6.7.3 finden Sie ein Beispiel für die Verwendung von PRESETON im Zusammenhang mit einer Anwendung Fliegendes Trennen.

---

**Hinweis**

Das Istwertsetzen ”PRESETON” darf nur mit dem Schlüsselwort ”WHEN” oder ”EVERY” erfolgen.

---

## 2.4.17 Mitschleppen und Kopplungen aktivieren, deaktivieren

**Einführung**

In der Funktionsbeschreibung:

**Literatur:** /FB/, M3, Mitschleppen

sind im Detail folgende Funktionen beschrieben.

- Mitschleppen
  Folgeachse(n) sind über einen Koppelfaktor mit einer Leitachse verbunden.
- Kurventabellen
  Kurventabellen stellen einen (komplexen) Zusammenhang zwischen Leitwert und Folgewert dar. Als Leitwert sind möglich:
  - von der Steuerung erzeugte Sollwerte
  - vom Geber ermittelte Istwerte
  - extern vorgegebene Größen

  Im Zusammenhang mit den Synchronaktionen ist besonders der Fall von Bedeutung, daß eine Folgeachse über Kurventabelle mit einer Leitachse verbunden wird.
- Leitwertkopplung
  Von den für Teileprogramme möglichen Leitwertkopplungen:
  - Achsleitwertkopplung
  - Bahnleitwertkopplung

  stehen für die Nutzung in Synchronaktionen nur Achsleitwertkopplungen zur Verfügung.

**Mitschleppen**

Aus einer Synchronaktion heraus kann die Zuordnung einer Folgeachse zu einer Leitachse mit einem Koppenfaktor definiert und gleichzeitig aktiviert werden:

... DO **TRAILON**(FA, LA, Kf)

mit:
FA Folgeachse
LA Leitachse
Kf Koppelfaktor

Die Auflösung des Mitschleppverbandes erfolgt mit:

... DO **TRAILOF**(FA, LA, LA2)

mit:
FA Folgeachse
LA Leitachse
LA2 Leitachse2, optional

**Kurventabellen**

Der in Kurventabellen hinterlegte Zusammenhang zwischen einer Leitwertgröße und einer Folgewertgröße kann in Synchronaktionen benutzt werden wie andere REAL–Funktionen (z.B. SIN, COS):

**Folgewert ermitteln**

Der sich aus einem Leitwert über die Kurventabelle n ergebende Folgewert soll einer Rechenvariablen zugewiesen werden.
Beispiel:

... DO $R17=**CTAB**(LW, n, grad)

mit:

| | |
|---|---|
| LW | Leitwert |
| n | Nummer der Kurventabelle |
| grad | Steigungsparameter, Ergebnis<br>(2 weitere opt. Parameter für Skalierung:<br>– Folgeachse<br>– Leitachse) |

Beispiel:
DEF REAL GRADIENT
...
WHEN $A_IN[1] == 1 DO $R17 = CTAB(75.0, 2, GRADIENT)

**Leitwert ermitteln**

Aus einer Synchronaktion heraus kann ein konkreter Leitwert anhand einer definierten Kurventabelle für einen Folgewert ermittelt werden.
Beispiel:

... DO $R18=**CTABINV**(FW, aprLW, n, grad)

mit:

| | |
|---|---|
| FW | Folgewert |
| aprLW | angenäherter Leitwert, mit dem bei mehrdeutiger Umkehrfunktion der Kurventabelle ein eindeutiger LW bestimmt werden kann |
| n | Nummer der Kurventabelle |
| grad | Steigungsparameter, Ergebnis<br>(2 weitere opt. Parameter für Skalierung:<br>– Folgeachse<br>– Leitachse) |

Die Funktionen CTAB und CTABINV sind sowohl in Bedingungen als auch im Aktionsteil von Synchronaktionen möglich.

**Achsleitwert kopplung**

Im Aktionsteil der Synchronaktion wird die Ankopplung der Folgeachse FA an die Leitachse LA über die gespeicherte Kurventabelle mit der Nummer NR wie folgt aufgerufen:

... DO **LEADON**(FA; LA, NR)
mit:

| | |
|---|---|
| FA | Folgeachse |
| LA | Leitachse |
| NR | Nummer der Kurventabelle |

**Achskopplung aus Synchronaktion ausschalten**

Soll die Achs–Leitwertkopplung beim Eintreffen einer weiteren Bedingung wieder aufgehoben werden, so lautet die Aktion:

... DO **LEADOF**(FA, LA)

**Systemvariablen**

Vom Teileprogramm und aus Synchronaktionen sind die Systemvariablen der Leitwertkopplung laut Liste der Systemvariablen lesbar/schreibbar.

Siehe 2.3.8.

**Erkennen des Synchronlaufes**

Die aus Teileprogramm und Synchronaktion lesbare Systemvariable $AA_SYNC[ax] zeigt an, ob und wie die Folgeachse FA synchronisiert ist:

0: nicht synchron

1: Synchronlauf grob (Gemäß MD 37200: COUPLE_POS_TOL_COARSE)

2: Synchronlauf fein (Gemäß MD 37210: COUPLE_POS_TOL_FINE)

**Anwendungsabgrenzung**

Im Teileprogramm direkt aktivierte Kopplungen werden an Satzgrenzen aktiviert. Mit der Möglichkeit der Kopplungsaktivierung durch Synchronaktionen wird eine ereignisgesteuerte differenzierte Aktivierung ermöglicht z.B.

– bei bestimmten Achsweg ab Satzanfang
– bei bestimmten Restweg bis Satzende
– Eintreffen von Digitaleingangssignalen
– Kombinationen aus diesen

S. 2.1, Bedingungen

Weitere Hinweise zu Programmierung der Kopplungsfunktionen und Kurventabellen finden Sie in:

**Literatur:** /PGA/, Programmieranleitung Arbeitsvorbereitung

---

**Hinweis**

Achsen, die beim Einkoppeln über Synchronaktionen in einem beliebigen Bewegungszustand angetroffen werden, werden durch die Steuerung synchronisiert. Details hierzu finden Sie in Funktionsbeschreibung M3.

---

**Beispiele**

In 6.7.3 finden Sie ein Beispiel, das von Achskopplung über Kurventabelle Gebrauch macht.

## 2.4.18 Messen aus Synchronaktionen

**Einführung**

Von den für Teileprogramme verfügbaren Meßfunktionen:
MEAS, MEAW, MEASA, MEAWA, MEAC

**Literatur:** /PGA/, Programmieranleitung Arbeitsvorbereitung
/FB/, M5, ”Messen”

stehen in Synchronaktionen nur folgende zur Verfügung:

- MEAWA achsiales Messen ohne Restweglöschen
- MEAC achsiales, kontinuierliches Messen

Während die Meßfunktion bei Bewegungssätzen im Teileprogramm jeweils auf einen Satz begrenzt ist, kann die Meßfunktion aus Synchronaktionen beliebig ein– und ausgeschaltet werden:

---

**Hinweis**

Mit statischen Synchronaktionen steht Messen auch in der Betriebsart JOG zur Verfügung.

---

**Programmierung**

**MEAWA**[Achse]=(Modus, Triggerereignis_1, Triggerereignis_2,
Triggerereignis_3, Triggerereignis_4)
; Achsiales Messen ohne Restweglöschen einschalten

**MEAC**[Achse]=(Modus, Meßspeicher,Triggerereignis_1, Triggerereignis_2,
Triggerereignis_3, Triggerereignis_4)
; achsiales, kontinuierliches Messen einschalten

Achse: Achse, für die gemessen wird

Tabelle 2-3 Modus–Bedeutungen:

| Zehnerdekade | Einerdekade | Bedeutung |
|---|---|---|
| | 0 | Meßauftrag abbrechen |
| | 1 | bis zu 4 **gleichzeitig** aktivierbare Triggerereignisse |
| | 2 | bis zu 4 **nacheinander** aktivierbare Triggerereignisse |
| | 3 | bis zu 4 **nacheinander** aktivierbare Triggerereignisse, jedoch keine Überwachung von Triggerereignis1 beim START |
| 0 | | aktives Meßsystem |
| 1 | | 1. Meßsystem |
| 2 | | 2. Meßsystem |
| 3 | | beide Meßsysteme |

Triggerereignis_1 bis Triggerereignis_4:

| | | |
|---|---|---|
| 1: | steigende Flanke Meßtaster 1 | |
| –1: | fallende Flanke Meßtaster 1 | *optional* |
| 2: | steigende Flanke Meßtaster 2 | *optional* |
| –2: | fallende Flanke Meßtaster 2 | *optional* |

Meßspeicher: Nummer einer FIFO–Variablen

Meßwerte werden ausschließlich für das **Maschinen**koordinatensystem bereitgestellt.

**MEAWA**

... DO **MEAWA**[Achse]=( , , , , ) ;Achsiales Messen ohne Restweglöschen

Restweglöschen kann bei Bedarf in der Synchronaktion explizit aufgerufen werden. S. 2.4.10 und Beispiel unten.

GEO–Achsen und an Transformationen beteiligte Achsen können einzeln programmiert werden.

Programmierung:
Die Programmierung entspricht der im Teileprogramm

---

**Hinweis**

Die Systemvariable $AC_MEA liefert für eine aus der Synchronaktion aufgerufene Messung keine auswertbare Information über die Gültigkeit der Messung.

Pro Achse darf nur ein Meßauftrag aktiv sein.

---

Systemvariablen:

| | | |
|---|---|---|
| $AA_MEAACT[Achse] | | liefert den augenblicklichen Meßstatus einer Achse. |
| | 1 | Messung aktiv |
| | 0 | Messung nicht aktiv |
| $A_PROBE[Meßtaster] | | liefert den momentanen Zustand des Meßtasters. |
| | 1 | Taster geschaltet, High Signal |
| | 0 | Taster nicht geschaltet, Low Signal |

Meßwerte im Maschinenkoordinatensystem mit 2 Meßtastern (Gebern):

| | |
|---|---|
| $AA_MM1[Achse] | Triggerereignis1, Geber 1 |
| $AA_MM2[Achse] | Triggerereignis 1, Geber 2 |
| $AA_MM3[Achse] | Triggerereignis 2, Geber 1 |
| $AA_MM4[Achse] | Triggerereignis 2, Geber 2 |

**MEAC**

... DO **MEAC**[Achse]=(Modus, Nr_FIFO, Triggerereignisse)

Die Variablen $AC_FIFO (S. 2.3.6.) sind dafür vorgesehen, Meßwerte aus zyklischen Meßvorgängen aufzunehmen. Modus und Triggerereignisse s.o.

Beispiele:
Für die folgenden Beispiele wurden per Maschinendaten 2 FIFO’s eingerichtet.

### Maschinendaten

```
MD 28050: MM_NUM_R_PARAM = 300
MD 28258: MM_NUM_AC_TIMER = 1
MD 28260: NUM_AC_FIFO = 2            ; 2 FIFO’s
MD 28262: START_AC_FIFO = 100        ; erster FIFO beginnt ab R100
MD 28264: LEN_AC_FIFO = 22           ; jeder FIFO kann 22 Werte aufnehmen
MD 28266: MODE_AC_FIFO = 0                   ; keine Summenbildung
```

Beispiel 1.
Auf einer Strecke zwischen X0 und X100 sollen alle steigenden Flanken von Meßtaster 1 aufgenommen werden. Es wird angenommen, daß nicht mehr als 22 Flanken auftreten können.

### Programm 1:

```
DEF INT ANZAHL
DEF INT INDEX_R
N0   G0 X0
N1   MEAC[X]=( 1, 1, 1) POS[X]=100
                              ; Modus = 1, gleichzeitig
                              ; Nr–FIFO       = 1
                              ; Triggerereignis 1= steigende Flanke, Meßgeber 1
N2   STOPRE                   ; Anhalten Vorverarbeitung
N3   MEAC[X]=( 0)             ; Abbrechen kontinuierliche Messung
N4   ANZAHL= $AC_FIFO1[4]     ; Anzahl eingetroffener Meßwerte in der FIFO–Variablen
N5   ANZAHL= ANZAHL – 1
N6   FOR INDEX_R= 0 TO ANZAHL
N7   R[INDEX_R]= $AC_FIFO1[0] ; FIFO–Inhalt in R0 – ... eintragen
N8   ENDFOR                   ; Nach Auslesen ist FIFO–Variable leer
```

Beispiel 2.
Auf einer Strecke zwischen X0 und X100 sollen alle steigenden und fallenden Flanken von Meßtaster 1 aufgenommen werden. Die Anzahl der erreichbaren Triggerereignisse ist unbekannt. Daraus folgt: Es müssen parallel in einer Synchronaktion die Meßwerte abgeholt und ab R1 aufsteigend abgelegt werden. Die Anzahl der abgelegten Meßwerte wird im R0 eingetragen.

### Programm 2:

```
N0   G0 X0                            ; Eilgang zum Startpunkt
N1   $AC_MARKER[1]=1                  ; Merker 1 als Index für Rechenvariablen R[..]
N2   ID=1 WHENEVER $AC_FIFO1[4]>=1
     DO $R[$AC_MARKER[1]]= $AC_FIFO1[0] $AC_MARKER[1]=$AC_MARKER[1]+1
                              ;Synchronaktion als Prüfung:
                              ; wenn 1 oder mehr Meßwerte in FIFO–Variable stehen
                              ; ältesten Wert aus FIFO auslesen und in aktuelle R[ ..]
                              ; ablegen, Index für R um 1 erhöhen
```

```
N3   MEAC[X]=( 1, 1, 1, –1)  POS[X]=100   ; Kontinuierliches Messen aktivieren, Bewegung
                                         ; nach X = 100
                                         ; Modus = 1, gleichzeitig
                                         ; Nr_FIFO = 1
                                         ; Triggerereignis 1= 1, steigende Flanke Meßgeber 1
                                         ; Triggerereignis 2= –1, fallende Flanke Meßgeber 1
N4   MEAC[X]=(0)                         ;  Messung abwählen
N5   STOPRE                              ; Vorverarbeitung stoppen
N6   R0= $AC_MARKER[1]                   ; Anzahl der erfaßten Werte in R0
```

Beispiel 3:
Kontinuierliches Messen mit explizitem Restweglöschen nach 10 Messungen

**Programm 3:**

```
N1   WHEN $AC_FIFO1[4]>=10
     DO  MEAC[X]=(0) DELDTG(X)           ; Schlußbedingung als Synchronaktion:
                                         ; Wenn 10 oder mehr Meßwerte in der FIFO–Variablen
                                         ; vorliegen,
                                         ; kontinuierliche Messung abwählen und
                                         ; Restweg löschen
N2   MEAC[X]=( 1,1,1,–1) G01 X100 F500   ; kontinuierliche Messung aus dem Teileprogramm aktiv.
                                         ; Modus = 1, gleichzeitig
                                         ; Nr_FIFO = 1, FIFO–Variable 1
                                         ; Triggerereignis 1= 1, steigende Flanke Meßgeber 1
                                         ; Triggerereignis 2= –1, fallende Flanke Meßgeber 1

N3   MEAC[X]=(0)                         ; Kontinuierliche Messung abwählen
N4   R0= $AC_FIFO1[4]                    ; tatsächliche Anzahl Meßwerte
```

**Priorität bei mehreren Messungen**

Zu einem Zeitpunkt kann pro Achse genau ein Meßauftrag aktiv sein.

Der Start eines Meßauftrags für dieselbe Achse bewirkt, daß die Trigger–ereignisse erneut aktiviert und die Meßergebnisse zurückgesetzt werden.
Wird Meßauftrag ausschalten (Modus 0) programmiert, ohne daß vorher ein Meßauftrag aktiviert wurde, so erfolgt keine gesonderte Reaktion.
Meßaufträge, die aus dem Teileprogramm gestartet wurden, können aus Synchronaktionen nicht beeinflußt werden.
Wird aus Synchronaktionen ein Meßauftrag für eine Achse gestartet, und für diese Achse ist bereits ein Meßauftrag aus dem Teileprogramm aktiv, so wird ein Alarm generiert.
Ist ein Meßauftrag aus Synchronaktionen aktiv, kann Messen aus dem Teileprogramm heraus nicht mehr gestartet werden.

**Meßaufträge und Zustandsänderungen**

Wenn der Meßauftrag aus Synchronaktionen erfolgt ist, zeigt die Steuerung folgende Verhalten:

| Zustand | Verhalten |
|---|---|
| Betriebsartenwechsel | Ein Meßauftrag, der durch eine modale Synchronaktion aktiviert wurde, wird durch den Betriebsartenwechsel nicht beeinflußt. Er bleibt über Satzgrenzen hinweg wirksam. |
| RESET | Der Meßauftrag wird abgebrochen |

| Zustand | Verhalten |
|---|---|
| Satzsuchlauf | Die Meßaufträge werden gesammelt und erst bei Erfüllung der programmierten Bedingung aktiviert |
| REPOS | Aktivierte Meßaufträge werden nicht beeinflußt. |
| Programmende | Meßaufträge, die aus statischen Synchronaktionen gestartet wurden, bleiben erhalten. |

### 2.4.19 Setzen und Löschen von Wartemarken der Kanalsynchronisation

**Einführung**

Die Koordination von Abläufen in den Kanälen ist beschrieben in:

**Literatur:** /FB/, K1, BAG, Kanal, Programmbetrieb

Von den dort genannten Funktionen sind die folgenden in Synchronaktionen zulässig:

**Wartemarke setzen**

Der Befehl **SETM**(MarkerNummer) kann im Teileprogramm und im Aktionsteil einer Synchronaktion gegeben werden. Er setzt die Marke MarkerNummer für den Kanal, in dem der Befehl läuft. (Eigener Kanal).

**Wartemarke löschen**

Der Befehl **CLEARM**(MarkerNummer) kann im Teileprogramm und im Aktionsteil einer Synchronaktion gegeben werden. Er löscht die Marke MarkerNummer für den Kanal, in dem der Befehl läuft. (Eigener Kanal).

### 2.4.20 Alarm setzen/ Fehlerreaktionen

**Fehlersituationen**

Alarm setzen ist eine Möglichkeit auf Fehlerzustände zu reagieren.

Anwendung:
Mit dem SETAL–Befehl können Zyklen–Alarme aus Synchronaktionen gesetzt werden.

Weitere Möglichkeiten auf Fehler zu reagieren sind:

- Achse stoppen S. 2.4.11
- Ausgang setzen S. 2.4.2
- Sonstige in 2.4 aufgeführte Aktionen

**Beispiel Alarm setzen**

```
ID=67 WHENEVER $AA_IM[X1] – $AA_IM[X2] < 4.567 DO SETAL(61000)

; Alarm setzen, wenn Abstand (Istwert der Achse X1 – Istwert der Achse X2)
; den kritischen Wert 4.567 unterschreitet.
```

**Zyklen und Zyklenalarme**

Hinweise zu Zyklen und Zyklenalarmen finden Sie in:

**Literatur:** /PGZ/, Programmieranleitung Zyklen

## 2.5 Aufruf von Technologiezyklen

**Definition**

Ein Technologiezyklus ist eine Folge von Aktionen, die sequentiell im Interpolationstakt abgearbeitet werden. Die in 2.4 dargestellten Aktionen können zu Programmen zusammengefaßt werden. Aus Anwendersicht handelt es sich bei diesen Programmen um Unterprogramme ohne Parameter.

**Parallelität im Kanal**

In einem Kanal können gleichzeitig mehrere Technologiezyklen oder Aktionen bearbeitet werden. Die Bearbeitung der Technologiezyklen und Aktionen des Kanals erfolgt parallel in einem Interpolationstakt.

**Unterschiede in der Bearbeitung**

Bezüglich der Bearbeitungsfolge muß der Anwender aus den folgenden Möglichkeiten die geeignetere wählen:

- Mehrere Aktionen in einer Synchronaktion:
  Die Aktionen werden alle gleichzeitig in dem Interpolationstakt ausgeführt, in dem die Bedingung erfüllt ist.
- Aktionen sind zu einem Technologiezyklus zusammengefaßt:
  Die Aktionen im Technologiezyklus werden im Interpolationstakt sequentiell abgearbeitet. Pro Interpolationstakt wird ein Satz abgearbeitet. Es müssen ein– und mehrtaktige Aktionen unterschieden werden. Ein Technologiezyklus ist dann beendet, wenn seine letzte Aktion ausgeführt ist ( in der Regel nach mehreren Interpolationstakten).

Befehle wie Variablenzuweisungen werden im Technologiezyklus in **einem** Interpolationstakt abgearbeitet. Andere Befehle (z.B. Bewegung einer Kommandoachse, S. 2.4.12) dauern **mehrere** Interpolationstakte. Ist die Funktion beendet (z.B. Genauhalt bei Positionieren einer Achse), so wird im darauffolgenden Interpolationstakt der nächste Satz ausgeführt.

Jeder Satz benötigt mindestens einen Interpolationstakt. Stehen mehrere eintaktige Aktionen in einem Satz, so werden diese in einem Interpolationstakt abgearbeitet. Bild 2-8 kennzeichnet beispielhaft, welche Aktionen eintaktig und welche mehrtaktig sind.

**Anwendung**

Mit Technologiezyklen ist es beispielweise möglich, jede Achse durch ein eigenes Achsprogramm zu bewegen.

**Programmierung**

In einer modalen/statischen Synchronaktion kann ein Technologiezyklus in Abhängigkeit von einer Bedingung aktiviert werden.

Das Programmende wird mit M02 / M17 / M30 / RET programmiert.

**Suchpfad**

Für den Aufruf gilt der Suchpfad wie bei Unterprogrammen und Zyklen.

Beispiel:

```
...
ID=1 EVERY $AA_IM[Y]>=10  DO AX_X          ; AX_X Unterprogramm–
                                           ; name für Achsprogramm für X–Achse
```

```
AX_X:                ; Achsprogramm
POS[X]=$R[7] FA[X]=377
$A_OUT[1]=1
POS[X]=R10
POS[X]=–90
M30
```

---

**Hinweis**

Ist die Bedingung erneut erfüllt, während der Technologiezyklus abgearbeitet wird, so wird er nicht von neuem gestartet. Wird der Technologiezyklus aus der Synchronaktion vom Typ WHENEVER gestartet und ist die Bedingung bei Ende des Technologiezyklus' noch erfüllt, so wird der Technologiezyklus von neuem gestartet.

---

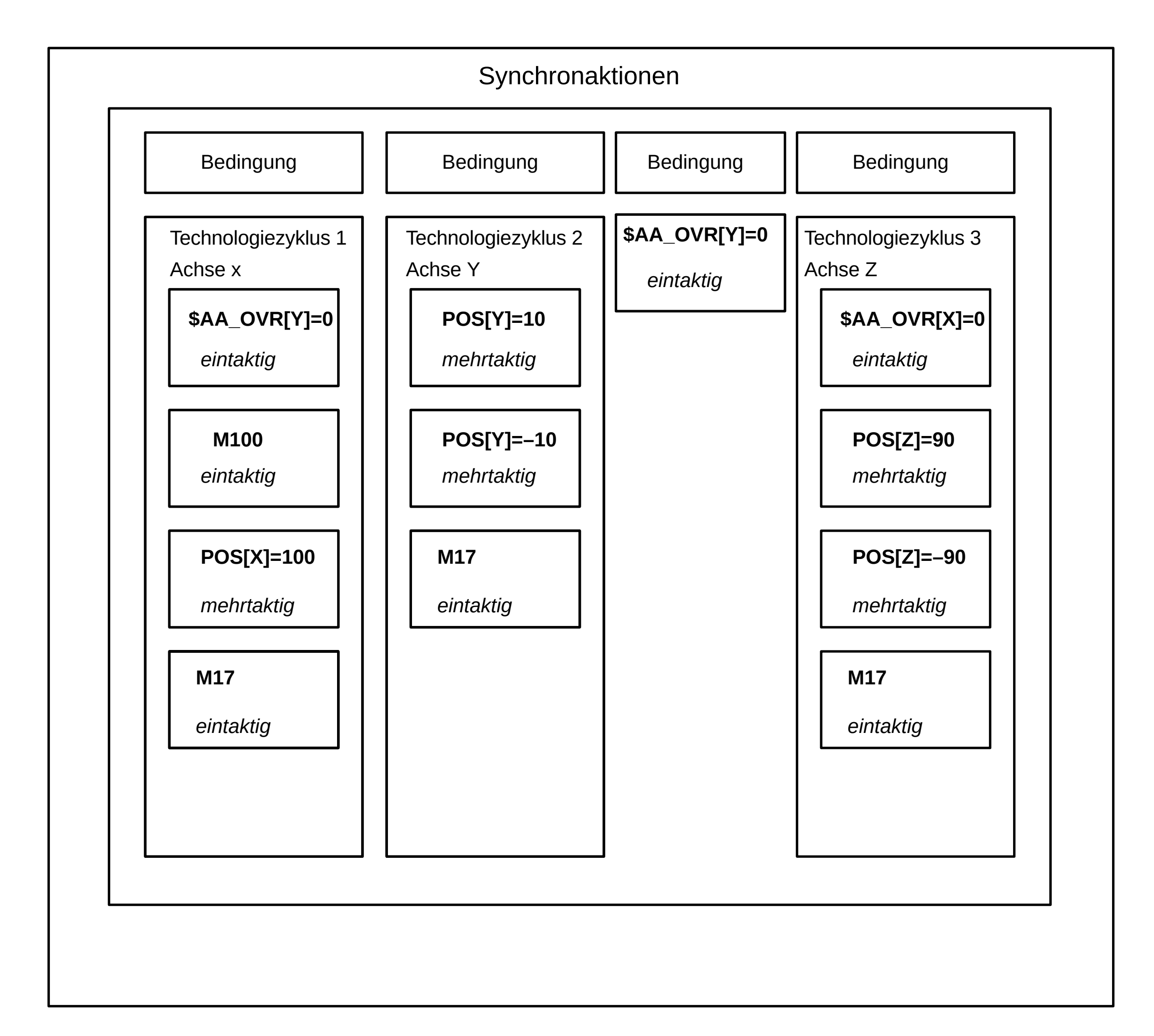



Bild 2-8 Mehrere Technologiezyklen

**Beispiel (2) für koordinierte Achsbewegungen:**
Durch Setzen digitaler NC–Eingänge werden verschiedene Achs–Programme gestartet.

Hauptprogramm:

```
...
ID=1 WHEN $A_IN[1]==1 DO ACHSE_X
ID=2 WHEN $A_IN[2]==1 DO ACHSE_Y
ID=3 WHEN $A_IN[3]==1 DO AA_OVR[Y]=0
ID=4 WHEN $A_IN[4]==1 DO ACHSE_Z
M30
```

Achsprogramme:

ACHSE_X:

```
$AA_OVR[Y]=0
M100
POS[X]=100
M17
```

ACHSE_Y:

```
POS[Y]=10
POS[Y]=–10
M17
```

ACHSE_Z:

```
$AA_OVR[X]=0
POS[Z]=90
POS[Z]=–90
M17
```

## 2.5.1 Koordinierungen zwischen Synchronaktionen, Technologiezyklen, Teileprogramm (und PLC)

**Beeinflussung von Technologiezyklen**

Die Technologiezyklen / Synchronaktionen werden über die Identifikationsnummer der Synchronaktionen beeinflußt, in denen sie als Aktion angegeben sind:

**Mittel zur Koordinierung**

| Schlüsselwort | Bedeutung | TP | SA |
|---|---|---|---|
| | Aufruf zulässig im Teileprogramm<br>Aufruf zulässig in Synchr. Aktion / Technologiezyklus | + | <br>+ |
| LOCK(ID) | Technologiezyklus sperren.<br>Die ggf. aktive Aktion wird unterbrochen. | | + |
| UNLOCK(ID) | Mit UNLOCK wird der Technologiezyklus an der Stelle der Unterbrechung fortgesetzt. Ein unterbrochener Positioniervorgang wird fortgesetzt. | | + |
| RESET(ID) | Technologiezyklus abbrechen. Aktive Positioniervorgänge werden abgebrochen. Wird der Technologiezyklus neu gestartet, so beginnt seine Bearbeitung mit dem 1. Satz im Technologiezyklus.<br>Je nach Synchronaktionstyp werden die Aktionen nach erneutem Eintreten der Bedingung wieder ausgeführt. Bereits ausgeführte Synchronaktionen vom WHEN–Typ werden nach RESET nicht mehr bearbeitet. | | + |
| CANCEL(ID) | Die Synchronaktion wird gelöscht. | + | |

- LOCK(ID), UNLOCK(ID) durch PLC s. 2.6.1

---

**Hinweis**

Eine Synchronaktion enthält einen Technologiezyklus–Aufruf. Weitere Aktionen sind in diesem Satz nicht erlaubt. Damit besteht Eindeutigkeit zwischen beeinflußter ID–Nummer und dem zugehörigen Technologiezyklus.

---

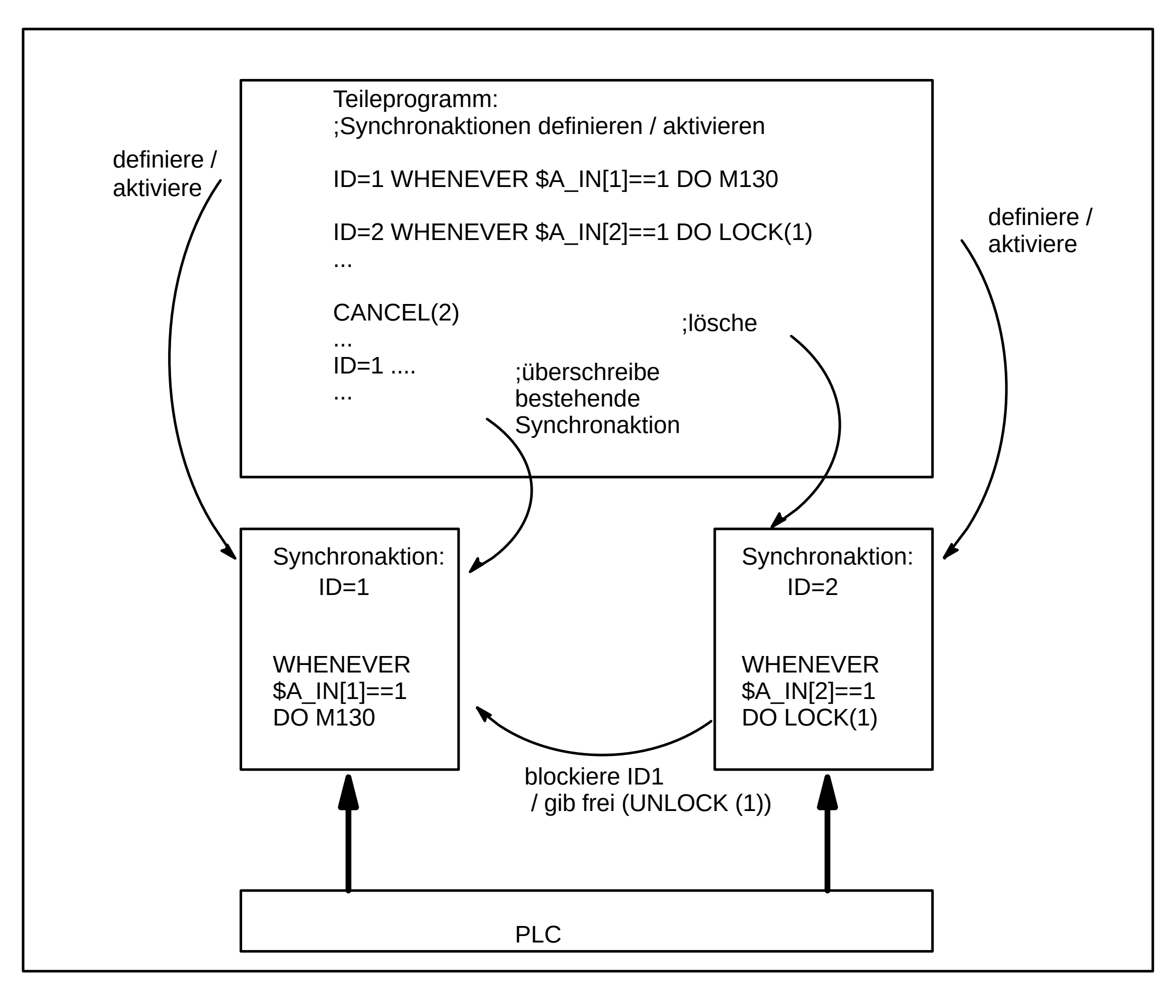



Bild 2-9 Anlegen / Verriegeln modaler Synchronaktionen / Löschen

## 2.6 Beeinflussung und Schutz von Synchronaktionen

### 2.6.1 Beeinflussung von PLC

**Funktion**

**Modale** Synchronaktionen (ID, IDS) können von PLC verriegelt bzw. freigegeben werden.

- Sperrung aller modalen Synchronaktionen
- Gezielte Sperrung einzelner Synchronaktionen

**Einflußbereich**

Die PLC kann auf maximal die ersten 64 modalen Synchronaktionen mit Sperren Einfluß nehmen (ID, IDS 1–64). Die durch PLC **sperrbaren** Synchronaktionen sind in einem 64 Bit großen Feld der Nahtstelle:
DB21–30, DBB308–315
durch die NC mit 1 gekennzeichnet. Geschützte Synchronaktionen sind nie als sperrbar gekennzeichnet. Siehe 2.6.2.

**Alle Synchronaktionen sperren**

Durch Setzen von DB 21–30, DBB1 Bit 2 durch das PLC–Anwendungsprogramm können alle modalen Synchronaktionen, die in der NC bereits definiert und gespeichert sind, von der Aktivierung ausgeschlossen werden. Eine Ausnahme bilden geschützte Synchronaktionen siehe 2.6.2.
Durch Setzen von DB 21–30, DBB1 Bit 2 auf 0 wird die pauschale Sperrung durch PLC wieder aufgehoben.

**Benutzung gezielte Sperren**

Für die ersten 64 IDs (1–64) ist in der PLC–Nahtstelle je ein Bit reserviert. (DB 21–30, DBB 300 Bit 0 bis DB21–30 DBB 307 Bit 7).

Standardmäßig sind die Funktionen freigegeben (Bits = 0). Durch Setzen des zugeordneten Bits wird die Auswertung der Bedingung und die Ausführung der dazugehörigen Funktion im NCK verriegelt.

**Aufhebung gezielter Sperren**

Durch Setzen des der ID–, IDS–Nummer entsprechenden Bits auf 0 in
DB 21–30, DBB 300 Bit 0 bis DB 21–30, DBB 307 Bit 7
wird eine zuvor gesperrte Synchronaktion wieder von PLC freigegeben.

**Aktualisieren der gezielten Sperren**

Wenn das PLC–Anwenderprogramm im Bereich von DB 21–30, DBB 300 Bit 0 bis DB 21–30, DBB 307 Bit 7 Änderungen vorgenommen hat, muß es diese mit DB 21–30 DBX280.1 aktivieren.

**Rückmeldung gezieltes Sperren**

Wenn von NCK die gezielten Sperren aktiviert wurden, wird dies in DB 21–30 DBX.281.1 signalisiert.

**Literatur:** /LIS/, Listen, Nahtstellensignale

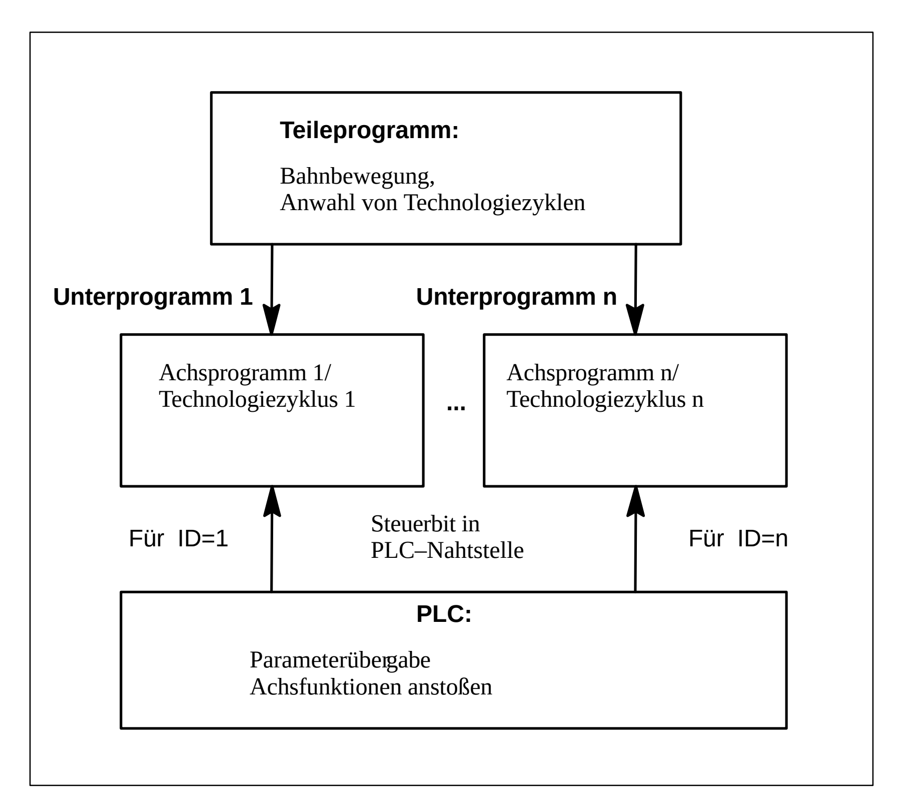



Bild 2-10 Achsprogramme/Technologiezyklen

**Lesen/Schreiben von PLC–Daten**

Ab SW–Stand 4 kann durch das Lesen / Schreiben von PLC–Daten aus dem Teileprogramm Parameterübergabe zwischen NCK und PLC über VDI–Nahtstelle erfolgen.
Dies ist eine Option: PLC–Variablen

**Literatur:** /FB/, P3, ”PLC–Grundprogramm”

Die Parameter sind auch aus Synchronaktionen zugänglich. Damit ist es möglich, vor Anstoß einer Achs–Funktion von PLC Daten zur Parametrierung an NCK zu geben. Die anzusprechenden Systemvariablen finden Sie in 2.3.8.

## 2.6.2 Geschützte Synchronaktionen

**Globaler Schutz**

**Funktion**

Über das Maschinendatum
MD 11500 : PREVENT_SYNACT_LOCK
kann ein Bereich von schreibgeschützten Synchronaktionen festgelegt werden. Synchronaktionen mit ID–Nummern, die im geschützten Bereich liegen, können **nicht** mehr:

– überschrieben,
– gelöscht (CANCEL) oder
– gesperrt (LOCK)

werden, wenn sie einmal definiert sind. Geschützte Synchronaktionen können auch durch PLC nicht gesperrt werden. Sie werden der PLC an der Nahtstelle als nicht sperrbar angezeigt. Vergleiche 2.6.1.

---

**Hinweis**

Die Funktionalität wird auch für Safety Integrated Systeme benutzt.

---

**Anwendungen**

Vom Maschinenhersteller definierte Reaktionen auf bestimmte Zustände sollen vom Endkunden nicht mehr beeinflußt werden können.

Die Inbetriebnahme beim Maschinenhersteller erfolgt noch ohne Schutz. Damit kann die Verknüpfungslogik definiert und getestet werden. Vor Auslieferung der Maschine erklärt der Maschinenhersteller den von ihm verwendeten Bereich von Synchronaktionen als geschützt. Damit ist es dem Endkunden nicht mehr möglich, eigene Synchronaktionen in diesem Bereich zu definieren.

**Notation des MD 11500**

```
$MN_PREVENT_SYNACT_LOCK[0]= i  ; i Nummer der 1. zu sperrenden ID
$MN_PREVENT_SYNACT_LOCK[1]= j  ; j Nummer der letzten zu sperrend. ID
```

i und j können auch vertauscht angegeben werden.
Mit i = 0 und j = 0 gibt es keinen geschützten Bereich.

### Kanalspezifischer Schutz

**Funktion**

Über das kanalspezifische Maschinendatum
MD 21240 : PREVENT_SYNACT_LOCK_CHAN
kann ein Bereich von schreibgeschützten Synchronaktionen für den Kanal festgelegt werden. Synchronaktionen mit ID–Nummern, die im geschützten Bereich liegen, können **nicht** mehr:

– überschrieben,
– gelöscht (CANCEL) oder
– gesperrt (LOCK)

werden, wenn sie einmal definiert sind. Geschützte Synchronaktionen können auch durch PLC nicht gesperrt werden. Sie werden der PLC an der Nahtstelle als nicht sperrbar angezeigt. Vergleiche 2.6.1.

**Anwendung**

s.o.

**Notation des MD 21240**

```
CHANDATA(C)    ; mit C Kanalnummer
$MC_PREVENT_SYNACT_LOCK_CHAN[0]= k
                ; k Nummer der 1. für den Kanal zu sperrenden ID
$MC_PREVENT_SYNACT_LOCK_CHAN[1]= l
                ; l Nummer der letzten für den Kanal zu sperrend. ID
```

k und l können auch vertauscht angegeben werden.
Mit k = 0 und l = 0 gibt es keinen geschützten Bereich.

Mit k = –1 und l = –1 wird angegeben, daß für den Kanal der mit
MD 11500 : PREVENT_SYNACT_LOCK festgelegte globale Bereich von geschützten Synchronaktionen gelten soll.

---

**Hinweis**

Während der Erstellung von geschützten statischen Synchronaktionen sollte der Schutz aufgehoben sein, da sonst bei jeder Änderung Power On notwendig ist, um die Logik neu definieren zu können.

---

Die Wirksamkeit der Sperren ist identisch, unabhängig davon, ob sie als:
globale Sperren oder als
kanalspezifische Sperren
angegeben wurden.

**Beispiel**

In einem System mit 2 Kanälen sollen Synchronaktionen wie folgt geschützt werden:
Im 1. Kanal sollen die IDs 20 bis 30 und
im Kanal 2 sollen die IDs 25 bis 35 geschützt werden. Es wird globale und kanalspezifische Angabe gemischt verwendet.

```
$MN_PREVENT_SYNACT_LOCK[0] = 25            ; globale Angabe

$MN_PREVENT_SYNACT_LOCK[1] = 35            ; globale Angabe

CHANDATA(1)

$MC_PREVENT_SYNACT_LOCK_CHAN[0] = 20
; im 1. Kanal wirkt nur das kanalspez. MD(1. zu schützende ID–Nummer)

$MC_PREVENT_SYNACT_LOCK_CHAN1] = 30
; im 1. Kanal wirkt nur das kanalspez. MD(letzte zu schützende ID–Nummer)

CHANDATA(2)

$MC_PREVENT_SYNACT_LOCK_CHAN[0] = –1
                 ; im 2. Kanal wirkt das globale Maschinendatum
                 ; $MN_PREVENT_SYNACT_LOCK!

$MC_PREVENT_SYNACT_LOCK_CHAN[1] = –1
...
```

## 2.7 Steuerungsverhalten für Synchronaktionen in bestimmten Betriebszuständen

### 2.7.1 Power On

Bei Power On sind keine Synchronaktionen aktiv. Statische Synchronaktionen, die sofort nach Power On aktiv sein sollen, müssen in einem von PLC gestarteten ASUP aktiviert werden.

**Literatur:** /FB/, P3, PLC–Grundprogramm
/FB/, K1, BAG, Kanal, Programmbetrieb

Voraussetzung dafür ist SW–Stand 4 mit der Funktionalität: ASUP in allen Betriebsarten.

Beispiele:

– AC–Regelung
– Safety Integrated, Verknüpfungslogik durch Synchronaktionen formuliert

### 2.7.2 RESET

**Bei Positionier–achsbewegungen**

Mit NC–Reset werden alle durch Synchronaktionen gestarteten Positionierbewegungen abgebrochen. Aktive Technologiezyklen werden zurückgesetzt.

**ID**

Programmlokale Synchronaktionen (mit ID=... programmiert) werden mit NC–Reset abgewählt.

**IDS**

Statische Synchronaktionen (mit IDS = ... programmiert) bleiben über NC–Reset hinaus erhalten. Aus diesen können nach NC–Reset wieder Bewegungen gestartet werden.

**Weitere Reaktionen, abhängig von Aktionen**

RESET Fortsetzung

| Synchronaktion / Technologiezyklus | **modale und satzweise**<br><br>aktive Aktion wird abgebrochen, Synchronaktionen werden gelöscht | **statisch (IDS)**<br><br>aktive Aktion wird abgebrochen, Technologiezyklus wird zurückgesetzt |
|---|---|---|
| **Achse / positionierende Spindel** | Bewegung wird abgebrochen | Bewegung wird abgebrochen |
| **drehzahlgeregelte Spindel** | $MA_SPIND_ACTIVE_AFTER_RESET== TRUE:<br>Spindel bleibt aktiv<br><br>$MA_SPIND_ACTIVE_AFTER_RESET==FALSE:<br>Spindel stoppt. | $MA_SPIND_ACTIVE_AFTER_RESET== TRUE:<br>Spindel bleibt aktiv<br><br>$MA_SPIND_ACTIVE_AFTER_RESET==FALSE:<br>Spindel stoppt. |
| **Leitwertkopplung** | $MC_RESET_MODE_MASK, Bit13 == 1:<br>Leitwertkopplung bleibt aktiv<br><br>$MC_RESET_MODE_MASK, Bit13 == 0:<br>Leitwertkopplung wird aufgelöst | $MC_RESET_MODE_MASK, Bit13 == 1:<br>Leitwertkopplung bleibt aktiv<br><br>$MC_RESET_MODE_MASK, Bit13 == 0:<br>Leitwertkopplung wird aufgelöst |
| **Messvorgänge** | aus Synchronaktionen gestartete Messvorgänge werden abgebrochen | aus statischen Synchronaktionen gestartete Messvorgänge werden abgebrochen |

### 2.7.3 NC–STOP

Aus statischen Synchronaktionen gestartete Bewegungen bleiben bei NC–Stopp aktiv.
Aus satzweisen und modalen Synchronaktionen gestartete Achsbewegungen werden unterbrochen und mit NC–Start fortgesetzt. Drehzahlgeregelte Spindeln bleiben aktiv
Die zum aktiven Satz gehörenden Synchronaktionen bleiben weiter aktiv.
Beispiel:
Ausgang setzen: ... DO $A_OUT[1] = 1

### 2.7.4 Betriebsartenwechsel

Es wird unterschieden zwischen programmlokalen und statischen Synchronaktionen.
Mit dem Schlüsselwort **IDS** aktivierte Synchronaktionen bleiben über Betriebsartenwechsel hinweg aktiv. Alle übrigen Synchronaktionen werden bei Betriebsartenwechsel inaktiv und mit dem Repositionieren bei Wechsel nach AUTO–Betrieb wieder aktiv.

Beispiel:

```
N10    WHEN $A_IN[1] == 1 DO DELDTG
N20    G1        X10 Y 200 F150 POS[U]=350
```

In Satz N20 wird gestoppt. Es erfolgt Betriebsartenwechsel nach JOG. War Restweglöschen vor der Unterbrechung noch nicht aktiv, so ist die im Satz N10 programmierte Synchronaktion nach der Rückkehr in Betriebsart AUTO und Fortsetzen des Programms weiter aktiv.

### 2.7.5 Programmende

Statische Synchronaktionen bleiben über Programmende hinaus aktiv.
Satzweise und modale Synchronaktionen werden abgebrochen.
Im M30–Satz wirken statische und modale Synchronaktionen weiter.
Sie können vor M30 mit CANCEL abgebrochen werden. Die mit FCTDEF programmierten Polynomkoeffizienten wirken über Programmende hinweg.

### 2.7.6 Verhalten der aktiven Aktionen bei Programmende und Betriebsartenwechsel

Siehe 2.7.4 und 2.7.5.

| **Synchronaktion / Technologiezyklus** | **modale und satzweise** werden abgebrochen | **statisch (IDS)** bleiben erhalten |
|---|---|---|
| **Achse / positionierende Spindel** | M30 wird verzögert, bis die Achse / Spindel steht. | Bewegung läuft weiter |
| **drehzahlgeregelte Spindel** | Programmende:<br>$MA_SPIND_ACTIVE_AFTER_RESET== TRUE:<br>Spindel bleibt aktiv<br><br>$MA_SPIND_ACTIVE_AFTER_RESET==FALSE:<br>Spindel stoppt<br><br>Bei Betriebsartenwechsel bleibt Spindel aktiv | Spindel bleibt aktiv |
| **Leitwertkopplung** | $MC_RESET_MODE_MASK, Bit13 == 1:<br>Leitwertkopplung bleibt aktiv<br><br>$MC_RESET_MODE_MASK, Bit13 == 0:<br>Leitwertkopplung wird aufgelöst | aus statischer Synchronaktion gestartete Kopplung bleibt erhalten |
| **Messvorgänge** | aus Synchronaktionen gestartete Messvorgänge werden abgebrochen | aus statischen Synchronaktionen gestartete Messvorgänge bleiben aktiv |

### 2.7.7 Satzsuchlauf

**Allgemein**

Die im Satzsuchlauf interpretierten Synchronaktionen des Programmes werden aufgesammelt. Die Bedingungen werden jedoch nicht ausgewertet. Aktionen werden nicht ausgeführt. Die Bearbeitung der Synchronaktionen beginnt erst mit NC–Start.

**IDS**

Mit dem Schlüsselwort IDS programmierte Synchronaktionen, die bereits aktiv sind, wirken auch während des Satzsuchlaufs.

**Polynom–koeffizienten**

Die mit FCTDEF programmierten Polynomkoeffizienten werden bei Satzsuchlauf **mit Berechnung** aufgesammelt, d.h. in die Systemvariablen geschrieben.

### 2.7.8 Programmunterbrechung durch ASUP

**ASUP–Anfang**

Modale und statische Bewegungssynchronaktionen bleiben erhalten und sind auch im asynchronen Unterprogramm wirksam.

**ASUP–Ende**

Wird das asynchrone Unterprogramm nicht mit Repos fortgesetzt, so wirken die im asynchronen Unterpogramm geänderten modalen und statischen Bewegungssynchronaktionen im Hauptprogramm weiter.
Aus Synchronaktionen gestartete Positionierbewegungen verhalten sich wie bei Betriebsartenwechsel:
Aus satzweisen und modalen Aktionen gestartete Bewegungen werden gestoppt und evtl. mit Repos fortgesetzt. Aus statischen Synchronaktionen gestartete Bewegungen laufen weiter.

### 2.7.9 REPOS

Im Restsatz gelten die Synchronaktionen wie im Unterbrechungssatz.
Änderungen an den modalen Synchronaktionen im asynchronen Unterprogramm sind im unterbrochenen Programm nicht wirksam.
Die mit FCTDEF programmierten Polynomkoeffizienten werden von ASUP und REPOS nicht beeinflußt.
Im asynchronen Unterprogramm wirken die Koeffizienten aus dem aufrufenden Programm. Im aufrufenden Programm wirken die Koeffizienten aus dem asynchronen Unterprogramm weiter.

Wurden mit Betriebsartenwechsel oder dem Start des Interruptprogramms Positionierbewegungen aus Synchronaktionen unterbrochen, so werden diese mit REPOS fortgesetzt.

### 2.7.10 Verhalten bei Alarmen

Über Synchronaktionen gestartete Achs– und Spindelbewegungen werden abgebremst, wenn ein Alarm mit Bewegungsstopp aktiv ist. Alle weiteren Aktionen (wie z.B. Ausgang setzen) werden weiter ausgeführt.

Löst eine Synchronaktion selbst einen Alarm aus, so wird diese Aktion im nächsten Interpolationstaktakt nicht weiter bearbeitet. Der Alarm wird also nur einmal abgesetzt. Alarme die Interpreterstopp als Alarmreaktion haben, wirken erst nach Abarbeiten der vordekodierten Sätze.

Alle weiteren Aktionen werden weiter bearbeitet.
Löst ein Technologiezyklus einen Alarm mit Bewegungsstopp aus, so wird der Technologiezyklus nicht weiter bearbeitet.

# 2.8 Projektierung

## 2.8.1 Projektierbarkeit

**Anzahl Synchron–aktionselemente**

Die Anzahl der programmierbaren Synchronaktionssätze hängt nur von der projektierbaren Anzahl von Synchronaktionselementen ab. Die Anzahl der Speicherelemente von Bewegungssynchronaktionen (Synchronaktionselementen) wird über das Maschinendatum
MD 28250: MM_NUM_SYNC_ELEMENTS
festgelegt.

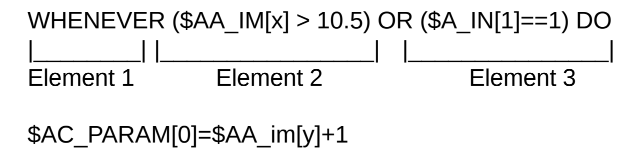
Die Festlegung ist unabhängig von der Anzahl der steuerungsintern verfügbaren Satzanzahl. Damit ist die Komplexität der in Echtzeit ausgewerteten Ausdrücke sowie die Anzahl der Aktionen flexibel einstellbar.

**Verwendung der Elemente**

Je **ein** Synchronaktionslement wird benötigt für:

– einen Vergleichsausdruck in der Bedingung
– eine elementare Aktion
– den Synchronaktionssatz

Beispiel:
Für den nachfolgenden Synchronaktionssatz werden insgesamt vier Elemente verbraucht.




$AC_PARAM[0]=$AA_im[y]+1
Element 4

Der Standardwert von MD 28250: $MC_MM_NUM_SYNC_ELEMENTS ist so eingestellt, daß die bis SW–Stand 3 fest vorgegebene Anzahl von max. 16 Synchronaktionen aktiviert werden kann.

---

**Hinweis**

Programmiert der Anwender keine Synchronaktionen, so kann er den Wert im MD 28250: MM_NUM_SYNC_ELEMENTS auf 0 setzen, um so ca. 16 KByte an DRAM Speicher einzusparen.

---

**Anzeige**

Mit der Statusanzeige für Synchronaktionen (S. Kapitel 2.9) läßt sich die Auslastung des Speichers für Synchronaktionen verfolgen oder aus Synchronaktionen über die Variable $AC_NUM_SYNC_ELEM lesen.

**Alarm**

Gehen die Elemente im Programmablauf aus, so wird ein Alarm abgesetzt. Der Anwender kann daraufhin die Anzahl der Synchronaktionselemente erhöhen oder sein Programm entsprechend abändern.

### Anzahl FCTDEF–Funktionen

Die Anzahl der Programmierbaren FCTDEF–Funktionen pro Satz wird über das Maschinendatum

MD 28252: MM_NUM_FCTDEF_ELEMENTS

projektiert.

Der Standardwert liegt für alle Steuerungstypen bei 3.
Den steuerungsabhängigen Maximalwert finden Sie in

**Literatur:** /LIS/, Listen.

### Interpolationstakt

Bei großer Anzahl von Synchronaktionen erhöht sich der Zeitbedarf für die Interpolationsebene. Der Interpolationstakt muß ggf. durch den Inbetriebnehmer dem Bedarf entsprechend verlängert werden.

### Richtwerte IPO–Takt Verlängerung

Als Orientierungshilfe werden einzelne Zeiten für Operationen innerhalb von Synchronaktionen (gemessen auf 840D mit NCU 573.x) angegeben:
Für andere Steuerungstypen sind Abweichungen möglich.

| **NC–Sprache** | **Zeitbedarf** | |
|---|---|---|
| | gesamt | **fett markierter Anteil** |
| Grundlast für eine Synchronaktion, wenn Bedingung nicht erfüllt ist:<br>**WHENEVER FALSE** DO $AC_MARKER[0]=0 | 10 µs | **~10 µs** |
| Variable lesen:<br>WHENEVER **$AA_IM[Y]>10** DO $AC_MARKER[0]=1 | 11 µs | **~1 µs** |
| Variable schreiben:<br>DO ***$R2=1*** | 11–12 µs | **~1–2 µs** |
| Settingdatum lesen / schreiben:<br>DO **$$SN_SW_CAM_MINUS_POS_TAB_1[0]=20** | 24 µs | **~14 µs** |
| Grundrechenarten, z.B. Multiplikation:<br>DO $R2=**$R2*2** | 22 µs | **~12 µs** |
| Trigonometrische Funktionen (z.B. cos):<br>DO $R2=**COS($R2)** | 23 µs | **~13 µs** |
| Positionierachs–Bewegung starten:<br>WHEN TRUE DO **POS[z]=10** | 83 µs | **~73 µs** |

## 2.9 Diagnose (nur mit MMC102/MMC103)

**Funktionalität der Diagnose**

Für die Diagnose von Synchronaktionen stehen die folgenden speziellen Testmittel zur Verfügung:

- Statusanzeige
- Es können aktuelle Werte aller Synchronaktions–Variablen angezeigt werden. (Echtzeitvariablen anzeigen)
- Es können Variablen–Verläufe im Interpolationstakt–Raster aufgezeichnet werden. (Echtzeitvariablen protokollieren)

Diese Funktionalität ist in der Bedienoberfläche wie folgt strukturiert:

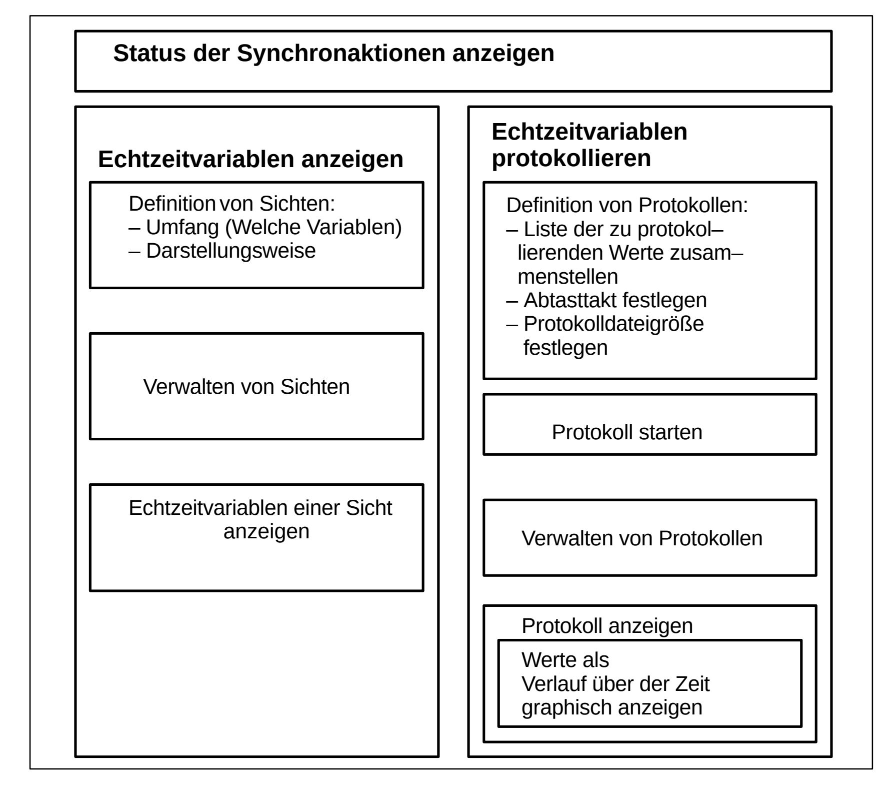



Bild 2-11 Funktionalität der Testmittel für Synchronaktionen

Die Beschreibung der Bedienung dieser Funktionen finden Sie in:

**Literatur:** /BA/, Bedienungsanleitung

## 2.9.1 Status der Synchronaktionen anzeigen

**Statusbild**

Das Statusbild zeigt an:

- Den aktuellen Ausschnitt des angewählten Programms

Alle programmierten Synchronaktionen nach:

- Zeilennummer
- Kennzeichen der Synchronaktionsart
- ID–Nummer der Synchronaktion (bei modalen Synchronaktionen)
- Status

**Synchronaktions-art**

Es werden unterschieden:

– ID Modale Synchronaktion

– IDS Statische modale Synchronaktion

– Satzweise Synchronaktion für den nächsten ausführbaren Satz ( nur im AUTOMATIK–Betrieb)

**Status**

Unter Status können auftreten:

- Keine Angabe: Bedingung wird im Interpolationstakt überprüft
- gesperrt Für die Synchronaktion wurde LOCK gesetzt
- aktiv Die Aktion läuft gerade ab. Besteht die Aktion aus einem Technologiezyklus, so wird zusätzlich die aktuelle Zeilennummer in diesem angezeigt.

**Vollständige Synchronaktionen**

Durch eine Suchfunktion kann zu jeder angezeigten Synchronaktion die ursprünglich programmierte Zeile in der NC–Sprache angezeigt werden.

## 2.9.2 Echtzeitvariablen anzeigen

Für den Test von Synchronaktionen ist es möglich, die Systemvariablen zu verfolgen. Die zulässigen Variablen werden in einer Vorschlagsliste zur Auswahl angeboten.
Die vollständige Liste der einzelnen Systemvariablen mit Kennzeichnung des Schreibzugriffs W und des Lesezugriffs R für Synchronaktionen finden Sie in:

**Literatur:** /PGA/, Programmieranleitung Arbeitsvorbereitung, Anhang

**Sichten**

In Sichten legt der Anwender fest, welche Werte für eine bestimmte Bearbeitungssituation wichtig sind und wie (nach Zeilen und Spalten, mit welchem Text) diese Werte angezeigt werden sollen. Es können mehrere Sichten zusammengestellt und in benannten Dateien abgespeichert werden.

**Sichten verwalten**

Eine definierte Sicht kann unter einem anwenderdefinierten Namen abgespeichert und wieder aufgerufen werden. Die in einer Sicht enthaltenen Variablen können verändert werden. (Sicht bearbeiten).

**Echtzeitvariable einer Sicht anzeigen**

Die Anzeige der zu einer Sicht gehörenden Werte erfolgt durch Aufruf der entsprechenden Anwendersicht.

## 2.9.3 Echtzeitvariablen protokollieren

**Ausgangssituation**

Die genaue Verfolgung der Abläufe in Synchronaktionen erfordert die Beobachtung der Zustände im Interpolationstakt.

**Methode**

Die in einer Protokolldefinition festgelegten Werte werden im angegebenen Takt in eine Protokolldatei definierter Größe eingeschrieben. Für die Anzeige der Inhalte der Protokolldateien werden Funktionen angeboten.

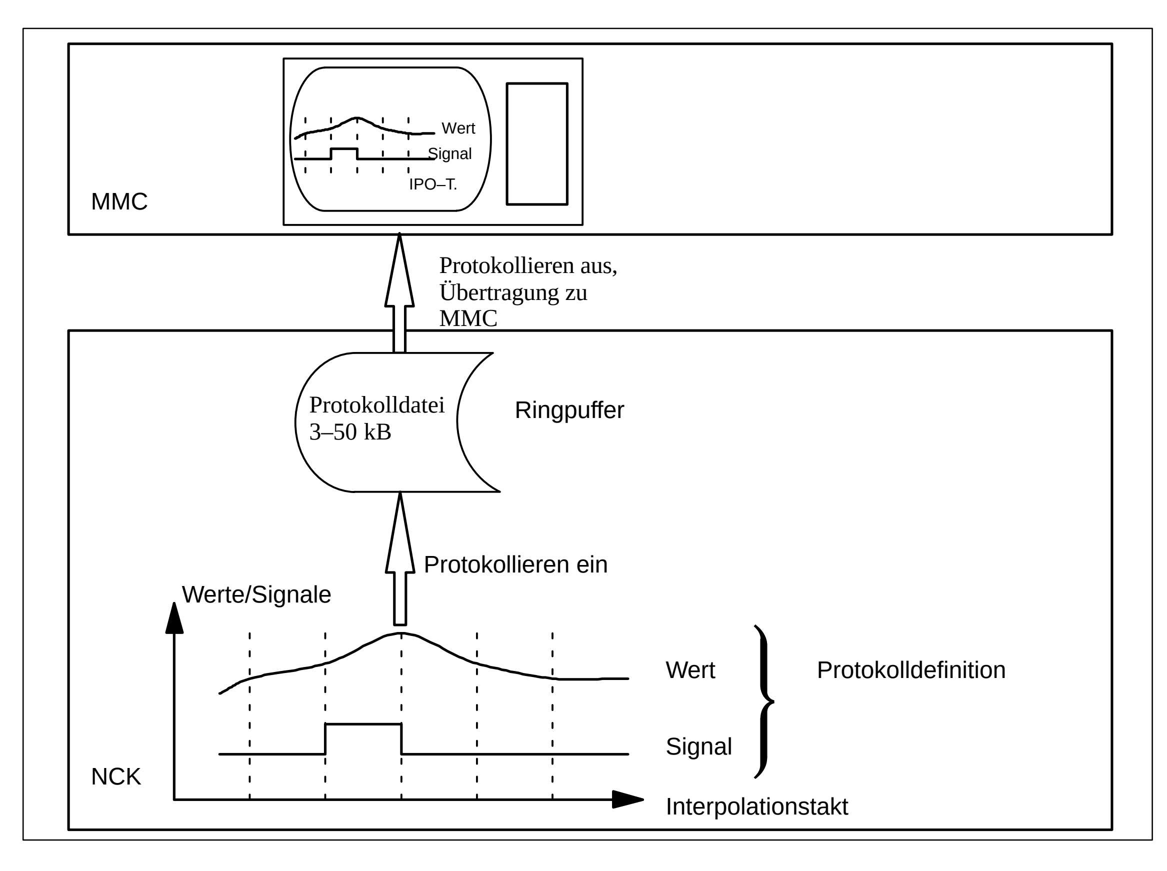



Bild 2-12 Schematischer Ablauf Echtzeitvariablen protokollieren

**Bedienung**

Die Hinweise zur Bedienung der Protokollierfunktion finden Sie in:

**Literatur:** /BA/, Bedienungsanleitung

**Protokolldefinition**

In der Protokolldefinition können bis zu 6 Variablen angegeben werden, deren Werte im angegebenen Takt in die Protokolldatei eingeschrieben werden sollen. Für die Auswahl der zu protokollierenden Variablen wird eine Liste angeboten. Der Takt ist in Vielfachen des Interpolationstaktes wählbar. Die Dateigröße in kB kann gewählt werden. Eine Protokolldefinition muß initialisiert werden, damit sie auf NCK aktiviert werden kann zum Erfassen der gewünschten Werte.

**Protokolldatei-größe**

Als Größe der Protokollierdatei können Werte von minimal 3 kB bis maximal 50 kB gewählt werden.

**Speichermethode**

Beim Überschreiten der effektiven Protokolldateigröße werden die ältesten Einträge überschrieben, so daß sich ein Ringpuffer ergibt.

**Protokollierung starten**

Die Protokollierung gemäß einer der initialisierten Protokolldefinitionen wird gestartet durch:

– Bedienung
– Setzen der Systemvariablen $A_PROTO=1 aus dem Teileprogramm

Der Startzeitpunkt muß so gewählt werden, daß die zu protokollierenden Variablen erst nach der Aktivierung von Abläufen auf der Maschine verändert werden. Der Start bezieht sich auf die zuletzt initialisierte Protokolldefinition.

**Protokollierung stoppen**

Die Funktion schließt die Protokolldatenerfassung im NCK ab. Die Datei mit den erfaßten Werten wird auf MMC zur Abspeicherung und Auswertung (Protokoll graphisch) bereitgestellt. Die Protokollierung kann gestoppt werden durch:

– Bedienung
– Setzen der Systemvariablen $A_PROTO=0 aus dem Teileprogramm

**Funktion Protokoll graphisch**

Die bis zu 6 Meßwerte eines Protokolls werden graphisch über der Abtastzeit dargestellt. Die Variablennamen werden in der Reihenfolge von oben nach unten entsprechend den Werteverläufen genannt. Die Platzverteilung auf dem Bildschirm erfolgt automatisch. Auf einen ausgewählten Teilbereich der Graphik kann eine Spreizung angewendet werden.

---

**Hinweis**

Die graphisch dargestellten Protokolle stehen auf MMC102 auch als Textdatei zur Verfügung. Mithilfe eines Editors können die exakten Werte eines Abtastzeitpunktes (Werte mit gleichem Zählindex) numerisch gelesen werden.

---

**Verwalten von Protokollen**

Es können mehrere Protokolldefinitionen unter anwenderdefinierten Namen gespeichert und für Initialisierung und Start der Aufzeichnung oder für Änderungen und Löschung wieder aufgerufen werden.

■

# 3 Randbedingungen

**Verfügbarkeit / Leistungsumfang**

Die möglichen Leistungen des Funktionspaketes Synchronaktionen hängen ab von:

- Dem Typ der SINUMERIK–Steuerung
  - HW
  - SW (Exportvariante / Standardvariante)
- Der Verfügbarkeit der durch ”Aktionen” auslösbaren Funktionen:
  - immer vorhandene Funktionen
  - als Option zu beziehende Funktionen

Die Leistungen der Steuerungen und ihrer Varianten und die als Optionen verfügbaren Funktionen sind beschrieben in den SW–Stand–spezifischen Katalogen:

**Literatur:** /BU/, Bestellunterlage, **Katalog NC60.1** und in
/LIS/, Listen

Darüberhinaus hängen die Funktionen der Synchronaktionen ab von:

- der Liste der aus Synchronaktionen lesbaren/änderbaren Systemvariablen einschließlich Maschinen– und Settingdaten.
  - Der Umfang hängt ab vom SW–Stand

Die für einen bestimmten SW–Stand nutzbaren Systemvariablen sind beschrieben in:

**Literatur:** /PGA/, Programmieranleitung Arbeitsvorbereitung, Anhang
(für den jeweiligen SW–Stand)

**Erweiterungen im SW–Stand 4**

Folgende Erweiterungen wurden mit SW–Stand 4 eingeführt:

- Diagnosemöglichkeiten für Synchronaktionen
- Verfügbarkeit zusätzlicher Echtzeitvariablen
- Komplexe Bedingungen in Synchronaktionen
  - Grundrechenarten
  - Funktionen
  - Indizierung mit Echtzeitvariablen
  - Zugriff auf Settingdaten und Maschinendaten
  - logische Operatoren
- Projektierbarkeit
  - Anzahl gleichzeitig aktiver Synchronaktionen
  - Anzahl spezieller Variablen für die Synchronaktionen
- Kommandoachsen/Achsprogramme/Technologiezyklen aus Synchronaktionen aktivieren

- PRESET aus Synchronaktionen
- Kopplungen und Mitschleppen aus Synchronaktionen
  - Einschalten
  - Ausschalten
  - Parametrieren
- Meßfunktionen benutzen aus Synchronaktionen
- SW–Nocken
  - Umdefinieren Position
  - Umdefinieren Vorhaltzeiten
- Restweglöschen ohne Vorlaufstopp
- Statische Synchronaktionen (andere Betriebsarten als AUTO möglich)
- Synchronaktionen:
  - schützen gegen Überschreiben und Löschen
  - anhalten, fortsetzen, löschen
  - Technologiezyklen rücksetzen
  - von PLC parametrieren, aktivieren, sperren
- Überlagerte Bewegung/Abstandsregelung verfeinert
- Kanalkoordination aus Synchronaktionen
- ASUP starten aus Synchronaktionen
- Hilfsfunktionsausgabe satzunabhängig
- alle erforderlichen Leistungen für Safety Integrated zur Formulierung der erforderlichen sicherheitsgerichteten logischen Verknüpfungen, geschützt gegen Veränderungen.
- 16 Synchronaktionen sind in der Grundausführung enthalten

**Erweiterungen im SW–Stand 5**

Folgende Leistungen erbringt SW 5 zusätzlich:

- Für PLC gekennzeichnete sperrbare Synchronaktionen
- Verfügbarkeit zusätzlicher Echtzeitvariablen
- Zugriff auf PLC–E/A (Option)
- mit der Option ”Synchronaktionen Stufe 2” sind 255 parallele Synchronaktionen pro Kanal möglich.
- Über das Programmende hinaus und in allen Betriebsarten wirkende statische Synchronaktionen IDS sind mit der Option ”Betriebsartübergreifende Aktionen, ASUPs und Synchronaktionen” möglich.

■

# Datenbeschreibungen (MD, SD) 4

## 4.1 Allgemeine Maschinendaten

<table>
<tr><td colspan="2">11500<br>MD–Nummer</td><td colspan="4">PREVENT_SYNACT_LOCK<br>Geschützte Synchronaktionen</td></tr>
<tr><td colspan="2">Standardvorbesetzung: 0, 0</td><td colspan="2">min. Eingabegrenze: 0</td><td colspan="2">max. Eingabegrenze: 255</td></tr>
<tr><td colspan="2">Änderung gültig nach Power On</td><td colspan="2">Schutzstufe: 2 / 7</td><td colspan="2">Einheit: –</td></tr>
<tr><td colspan="3">Datentype: DWORD</td><td colspan="3">gültig ab SW–Stand: 4.1</td></tr>
<tr><td colspan="2">Bedeutung:</td><td colspan="4">Erste und letzte ID eines geschützten Synchronaktions–Bereichs.<br>Synchronaktionen mit IDs in diesem Bereich können nicht überschrieben oder im Programm gesperrt (NC: CANCEL, LOCK) werden. Über PLC können die geschützten Synchronaktionen auch nicht gesperrt (LOCK) werden.<br>Typische Anwendung: Der Maschinenhersteller definiert eine Sicherheitslogik in einem asynchronen Unterprogramm. Dieses wird bei Power On von PLC gestartet. Der Bereich der verwendeten IDs wird über das Maschinendatum gesperrt. Damit hat der Endkunde keine Möglichkeit, die vom Maschinerhersteller definierte Logik zu verändern oder außer Kraft zu setzen.<br>Hinweis: Während der Erstellung der zu schützenden Synchronaktionen sollte der Schutz aufgehoben werden, da sonst bei jeder Änderung Power On notwendig ist, um die Logik neu definieren zu können.<br>Mit 0,0 gibt es keinen Bereich von geschützten Synchronaktionen. Die Funktion ist ausgeschaltet. Die Werte werden als Absolutwerte gelesen und Ober– und Unterwert können in beliebiger Reihenfolge angegeben werden.<br>Die Projektierung kann durch das kanalspezifische MD 21240: PREVENT_SYNACT_LOCK_CHAN ggf. umprojektiert werden.</td></tr>
<tr><td colspan="2">korrespondierend mit ....</td><td colspan="4">MD 21240: PREVENT_SYNACT_LOCK_CHAN</td></tr>
</table>

## 4.2 Kanalspezifische Maschinendaten

| **21240**<br>MD–Nummer | **PREVENT_SYNACT_LOCK_CHAN**<br>Geschützte Synchronaktionen des Kanals | |
|---|---|---|
| Standardvorbesetzung: –1, –1 | min. Eingabegrenze: –1 | max. Eingabegrenze: 255 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 6.4 | |
| Bedeutung: | Erste und letzte ID eines geschützten Synchronaktions–Bereichs.<br>Synchronaktionen mit IDs in diesem Bereich können nicht überschrieben oder im Programm gesperrt (NC: CANCEL, LOCK) werden. Über PLC können die geschützten Synchronaktionen auch nicht gesperrt (LOCK) werden.<br>Der Bereich der verwendeten IDs wird über das Maschinendatum gesperrt. Damit hat der Endkunde keine Möglichkeit, die vom Maschinerhersteller definierte Logik zu verändern oder außer Kraft zu setzen.<br>Hinweis: Während der Erstellung der zu schützenden Synchronaktionen sollte der Schutz aufgehoben werden, da sonst bei jeder Änderung Power On notwendig ist, um die Logik neu definieren zu können.<br>Mit 0,0 gibt es keinen Bereich von geschützten Synchronaktionen. Die Funktion ist ausgeschaltet. Die Werte werden als Absolutwerte gelesen und Ober– und Unterwert können in beliebiger Reihenfolge angegeben werden.<br>Mit –1, –1 wird angegeben, daß für den Kanal die mit<br>MD 11500: PREVENT_SYNACT_LOCK festgelegten ID–Nummern gelten sollen. | |
| korrespondierend mit .... | MD 11500: PREVENT_SYNACT_LOCK | |

| **28250**<br>MD–Nummer | **MM_NUM_SYNC_ELEMENTS**<br>Anzahl Elemente für Ausdrücke der Synchronaktionen | |
|---|---|---|
| Standardvorbesetzung: 159 | min. Eingabegrenze: 0 | max. Eingabegrenze: 2000 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Die Teile der Bewegungssynchronaktionen werden für die Abspeicherung in der Steuerung in Speicherelementen abgelegt. Eine Bewegungssynchronaktion belegt minimal 4 Elemente. Es belegen:<br>– jeder Operand in der Bedingung 1 Element<br>– jede Aktion >= 1 Element<br>– jede Zuweisung 2 Elemente<br>– jeder weitere Operand in komplexen Ausdrücken 1 Element.<br>Ein Element belegt ca 64 Bytes. | |
| weiterführende Literatur | Programmieranleitung Arbeitsvorbereitung | |

| **28252**<br>MD–Nummer | **MM_NUM_FCTDEF_ELEMENTS**<br>Anzahl der FCTDEF–Elemente | |
|---|---|---|
| Standardvorbesetzung: 3 | min. Eingabegrenze: 0 | max. Eingabegrenze: 100 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Für die Abspeicherung von Funktionen in der Steuerung für die Verwendung in Synchronaktionen werden Speicherelemente benötigt. Das MD legt die Anzahl dieser Elemente fest. | |

| **28254**<br>MD–Nummer | **MM_NUM_AC_PARAM**<br>Parameteranzahl $AC_PARAM | |
|---|---|---|
| Standardvorbesetzung: 50 | min. Eingabegrenze: 0 | max. Eingabegrenze: 10000,<br>ab SW 6.3: 20000 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Anzahl kanalspezifischer Parameter $AC_PARAM für Bewegungssynchronaktionen | |

| **28255**<br>MD–Nummer | **MM_BUFFERED_AC_PARAM**<br>Speicherort für $AC_PARAM | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 1 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 6.3 | |
| Bedeutung: | Die Systemvariablen $AC_PARAM können wahlweise abgespeichert werden:<br>0: im dynamischen Speicher DRAM, Vorbesetzung<br>1: im statischen Speicher SRAM<br>Im SRAM gespeicherte Systemvariablen behalten über RESET und Power On hinweg ihre aktuellen Werte. Sie können in die Datensicherung einbezogen werden. | |
| korrespondierend mit .... | MM_NUM_AC_PARAM | |
| weiterführende Literatur | /IAD/, Inbetriebnahmeanleitung | |

| **28256**<br>MD–Nummer | **MM_NUM_AC_MARKER**<br>Merkeranzahl $AC_MARKER | |
|---|---|---|
| Standardvorbesetzung: 8 | min. Eingabegrenze: 0 | max. Eingabegrenze: 10000,<br>ab SW 6.3: 20000 |
| Änderung gültig nach POwer On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Anzahl kanalspezifischer Merker $AC_MARKER für Bewegungssynchronaktionen | |

| **28257**<br>MD–Nummer | **MM_BUFFERED_AC_MARKER**<br>Speicherort für $AC_MARKER | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 1 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: | gültig ab SW–Stand: 6.3 | |
| Bedeutung: | Die Systemvariablen $AC_MARKER können wahlweise abgespeichert werden:<br>0: im dynamischen Speicher DRAM, Vorbesetzung<br>1: im statischen Speicher SRAM<br>Im SRAM gespeicherte Systemvariablen behalten über RESET und Power On hinweg ihre aktuellen Werte. Sie können in die Datensicherung einbezogen werden. | |
| korrespondierend mit .... | MM_NUM_MARKER | |
| weiterführende Literatur | /IAD(, Inbetriebnahmeanleitung | |

| **28258**<br>MD–Nummer | **MM_NUM_AC_TIMER**<br>Anzahl Zeitvariablen $AC_TIMER | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 10000 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Anzahl kanalspezifischer Zeitvariablen $AC_TIMER für Bewegungssynchronaktionen | |

| **28260**<br>MD–Nummer | **NUM_AC_FIFO**<br>Anzahl Variablen $AC_FIFO1, $AC_FIFO2, ... | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 10 |
| Änderung gültig nach Power On | Schutzstufe: 2 / / | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Anzahl FIFO–Variablen $AC_FIFO1 bis $AC_FIFO10 für Bewegungssynchronaktionen. | |
| Anwendungsbeispiel(e) | FIFO–Variable dienen z.B. zur Produktverfolgung: In jeder FIFO–Variable kann für jedes Teil auf einem Band eine Information (z.B. die Produktlänge) zwischengespeichert werden. | |
| korrespondierend mit .... | MD 28262: START_AC_FIFO | |

| **28262**<br>MD–Nummer | **START_AC_FIFO**<br>FIFO–Variablen speichern ab R–Parameter | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 10000 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Nummer des R–Parameters, ab dem FIFO–Variablen gespeichert werden.<br>Alle R–Parameter mit niedrigeren Nummern können beliebig im Teileprogramm verwendet werden.<br>R–Parameter oberhalb des FIFO–Bereichs können aus dem Teileprogramm nicht beschrieben werden.<br>Die Anzahl der R–Parameter muß über das Maschinendatum<br>MD 28050: $MC_MM_NUM_R_PARAM so eingestellt werden, daß ab dem<br>Start R–Parameter alle FIFO–Variable untergebracht werden können:<br>$MC_MM_NUM_R_PARAM=$MC_START_FIFO +<br>$MC_NUM_AC_FIFO*($MC_LEN_AC_FIFO+6)<br>Die FIFO–Variable tragen die Namen $AC_FIFO1 bis $AC_FIFOn.<br>Sie sind als Felder angelegt.<br>Die Indizes 0 – 5 haben Sonderbedeutungen:<br>n= 0: Beim Schreiben mit Index 0 wird ein neuer Wert in den FIFO abgelegt<br>Beim Lesen mit Index 0 wird das älteste Element gelesen und aus dem FIFO entfernt<br>n=1: Zugriff auf das zuerst eingelesene Element<br>n=2: Zugriff auf das zuletzt eingelesene Element<br>n=3: Summe aller FIFO–Elemente<br>n=4: Anzahl der im FIFO verfügbaren Elemente<br>n=5: aktueller Schreibindex relativ zum FIFO–Anfang | |
| korrespondierend mit .... | MD 28260: NUM_AC_FIFO | |

| **28264**<br>MD–Nummer | **LEN_AC_FIFO**<br>Länge der FIFO–Variablen $AC_FIFO ... | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 10000 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – |
| Datentype: DWORD | gültig ab SW–Stand: 4.1 | |
| Bedeutung: | Länge der FIFO–Variablen $AC_FIFO1 bis $AC_FIFO10.<br>Alle FIFO–Variablen eines Kanals haben gleiche Länge. | |
| korrespondierend mit .... | MD 28262, MD 28260 | |

<table>
<tr><td>28266<br>MD–Nummer</td><td colspan="3">MODE_AC_FIFO<br>Modus der FIFO–Bearbeitung</td></tr>
<tr><td>Standardvorbesetzung: 0</td><td>min. Eingabegrenze: 0</td><td colspan="2">max. Eingabegrenze: ***</td></tr>
<tr><td colspan="2">Änderung gültig nach Power On</td><td>Schutzstufe: 2 / 7</td><td>Einheit: –</td></tr>
<tr><td colspan="2">Datentype: BYTE</td><td colspan="2">gültig ab SW–Stand: 4.1</td></tr>
<tr><td>Bedeutung:</td><td colspan="3">Modus der FIFO–Bearbeitung:<br>Bit 0 = 1: Die Summe aller FIFO–Inhalte wird bei jedem Schreibzugriff aktuell gebildet.<br>Bit 0 = 0: Keine Summenbildung</td></tr>
<tr><td>korrespondierend mit ....</td><td colspan="3">MD 28260: NUM_AC_FIFO</td></tr>
</table>

## 4.3 Achs–/Spindelspezifische Maschinendaten

| **30450**<br>MD–Nummer | **IS_CONCURRENT_POS_AX**<br>Konkurrierende Positionierachse | |
|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 1 |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: 1 |
| Datentype: Boolean | gültig ab SW–Stand: 1 | |
| Bedeutung: | Bei der Achse handelt es sich um eine konkurrierende Positionierachse.<br>AB SW4.3 (nicht FM–NC):<br>Wenn FALSE: Bei RESET wird eine neutrale Achse wieder Kanalachse.<br>Wenn TRUE: Bei RESET bleibt eine neutrale Achse im Zustand neutrale Achse, und eine Kanalachse wird neutrale Achse | |
| weiterführende Literatur | Starten von kommandoachsen S. 2.4.12 | |

| **32070**<br>MD–Nummer | **CORR_VELO**<br>Achsgeschwindigkeit für Handrad, ext. NPV, cont. Dressing, Abstandsregelung | |
|---|---|---|
| Standardvorbesetzung: 100 | min. Eingabegrenze: 0 | max. Eingabegrenze: plus |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: % |
| Datentype: DWORD | gültig ab SW–Stand: 3.2 | |
| Bedeutung: | Begrenzung der Achsgeschwindigkeit für Handradüberlagerung, externe Nullpunktverschiebung, Continuous Dressing, Abstandsregelung $AA_OFF über Synchronaktionen bezogen auf die JOG–Geschwindigkeit MD: JOG_VELO, MD: JOG_VELO_RAPID, MD: JOG_REV_VELO, MD: JOG_REV_VELO_RAPID.<br>Die maximal zulässige Geschwindigkeit ist die maximale Geschwindigkeit im MD: MAX_AX_VELO. Auf diesen Wert wird begrenzt. Eine Überschreitung dieses Werts wird durch Alarm angezeigt.<br>Die Umrechnung nach Linear– oder Rundachsgeschwindigkeit erfolgt entsprechend MD: IS_ROT_AX. | |
| Anwendungsbeispiel(e) | Begrenzung der Geschwindigkeit beim Verfahren überlagerter Bewegungen. | |

| 32074<br>MD–Nummer | FRAME_OR_CORRPOS_NOTALLOWED<br>Wirksamkeit der Frames und Werkzeuglängenkorrektur | | |
|---|---|---|---|
| Standardvorbesetzung: 0 | min. Eingabegrenze: 0 | max. Eingabegrenze: 0xFF | |
| Änderung gültig nach Power On | Schutzstufe: 2 / 7 | Einheit: – | |
| Datentype: DWORD | gültig ab SW–Stand: 4.2 | | |
| Bedeutung: | Über dieses Maschinendatum wird die Wirksamkeit der Frames und Werkzeuglängenkorrekturen für Teilungsachsen, PLC–Achsen und aus Synchronaktionen gestartete Kommandoachsen festgelegt.<br><br>Bit == 0: Frame bzw. Korrekturwerte sind erlaubt<br><br>Bitbelegung:<br>Bit 0 == 1: programmierbare Nullpunktverschiebung (TRANS) für Teilungsachse verboten.<br><br>Bit 1 == 1: Maßstabsänderung (SCALE) für Teilungsachse verboten<br><br>Bit 2 == 1: Richtungsumkehr (MIRROR) für Teilungsachse verboten<br><br>Bit 3 == 1: DRF–Verschiebung für Achse verboten<br><br>Bit 4 == 1: Externe Nullpunktverschiebung für Achse verboten<br><br>Bit 5 == 1: Online–Werkzeugkorrektur für Achse verboten<br><br>Bit 6 == 1: Synchronaktions–Offset für Achse verboten<br><br>Bit 7 == 1: Compilezyklen–Offset für Achse verboten<br><br>Bit 8 == 1: axiale Frames werden für PLC–Achsen berücksichtigt<br>Bit 8 == 0: axiale Frames werden für PLC–Achsen NICHT berücksichtigt (Bitauswertung so aus Kompatibilitaetsgründen)<br><br>Bit 9 == 1: axiale Frames werden für Kommandoachsen NICHT berücksichtigt<br>Bit 9 == 0: axiale Frames werden für Kommandoachsen berücksichtigt | | |

| 32920<br>MD–Nummer | AC_FILTER_TIME<br>Filter–Glättungskonstante für Adaptive–Control | | |
|---|---|---|---|
| Standardvorbesetzung: 0.0 | min. Eingabegrenze: 0.0 | max. Eingabegrenze: plus | |
| Änderung gültig nach Power On | Schutzstufe: 2/7 | Einheit: s | |
| Datentype: DOUBLE | gültig ab SW–Stand: 2.1 | | |
| Bedeutung: | Mit den Hauptlaufvariablen $AA_LOAD, $AA_POWER, $AA_TORQUE und $AA_CURR können die folgenden Antriebs–Istwerte erfasst werden:<br>– Antriebsauslastung<br>– Antriebswirkleistung<br>– Antriebsmomentensollwert<br>– Stromistwert der Achse oder Spindel<br><br>Um Spitzen auszugleichen, können die gemessenen Werte durch ein PT1–Filter geglättet werden. Die Filterzeitkonstante wird mit dem MD: AC_FILTER_TIME (Filter–Glättungszeitkonstante für Adaptive–Control) definiert.<br><br>Bei Erfassung des Antriebsmomentensollwerts oder Stromistwerts wirkt das Filter zusätzlich zu den im 611–D vorhandenen Filtern. Beide Filter werden hintereinander geschaltet, wenn im System sowohl stark wie auch schwach geglättete Werte benötigt werden. Durch Vorgabe der Glättungszeit 0 Sekunden wird das Filter ausgeschaltet. | | |
| MD irrelevant bei ...... | FM–NC mit 611A | | |
| Anwendungsbeispiel(e) | Glättung des Stromistwertes bei AC–Regelung. | | |

<table>
<tr><td>36750<br>MD–Nummer</td><td colspan="3">AA_OFF_MODE<br>Wirkung der Wertzuweisung für axiale Überlagerung bei Synchronaktionen</td></tr>
<tr><td>Standardvorbesetzung: 0</td><td>min. Eingabegrenze: 0</td><td colspan="2">max. Eingabegrenze: 7</td></tr>
<tr><td colspan="2">Änderung gültig nach Power On</td><td>Schutzstufe: 2/7</td><td>Einheit: –</td></tr>
<tr><td colspan="2">Datentype: BYTE</td><td colspan="2">gültig ab SW–Stand: 3.2 (ab SW 6 Bit 1 und 2)</td></tr>
<tr><td>Bedeutung:</td><td colspan="3">Mit der Hauptlaufvariablen $AA_OFF kann eine überlagerte Bewegung für die programmierte Achse innerhalb einer Synchronaktion realisiert werden.<br>Über das achsiale MD: AA_OFF_MODE wird die Art der Verrechnung wie folgt definiert:<br><br>Bit0: Wirkung der Werkzeugzuweisung innerhalb einer Synchronvariablen: ab SW 3.2<br>Bit0 = 0: absoluter Wert<br>Bit0 = 1: inkrementeller Wert (Integrator)<br><br>Bit1: Verhalten von $AA_OFF bei RESET<br>Bit1 = 0: $AA_OFF wird bei RESET abgewählt<br>Bit1 = 1: $AA_OFF bleibt über RESET hinaus erhalten (ab SW 6)<br><br>Bit2: $AA_OFF in der Betriebsart JOG<br>Bit2 = 0: keine Überlagerte Bewegung aufgrund von $AA_OFF<br>Bit2 = 1: Überlagerte Bewegung wird aufgrund von $AA_OFF interpoliert (ab SW 6)</td></tr>
<tr><td>Anwendungsbeispiel(e)</td><td colspan="3">• Abstandsregelung für Laserbearbeitung (integrierend)<br>• Joystick gesteuertes Achsverfahren (proportional)</td></tr>
</table>

## 4.4 Settingdaten

<table>
<tr><td>43350<br>MD–Nummer</td><td colspan="3">AA_OFF_LIMIT<br>Obergrenze des Korrekturwertes für $AA_OFF Abstandsregelung</td></tr>
<tr><td>Standardvorbesetzung: 1.0 Ex+8</td><td>min. Eingabegrenze: 0</td><td colspan="2">max. Eingabegrenze: ***</td></tr>
<tr><td colspan="2">Änderung gültig nach SOFORT</td><td>Schutzstufe: 2 / 7</td><td>Einheit: mm/ Grad</td></tr>
<tr><td colspan="2">Datentype: DOUBLE</td><td colspan="2">gültig ab SW–Stand: 4.2</td></tr>
<tr><td>Bedeutung:</td><td colspan="3">Obergrenze des Korrekturwertes, der über Synchronaktionen über die Variable $AA_OFF vorgegeben werden kann.<br>Der Grenzwert wirkt auf den absolut wirksamen Korrekturbetrag.<br>Anwendung für die Abstandsregelung bei Laserbearbeitung: Der Korrekturwert wird begrenzt, damit sich der Laser–Kopf nicht in Blechausschnitten verhaken kann.<br>Über die Systemvariable $AA_OFF_LIMIT kann abgefragt werden, ob sich der Korrekturwert im Grenzbereich befindet.</td></tr>
</table>

■

# Signalbeschreibungen



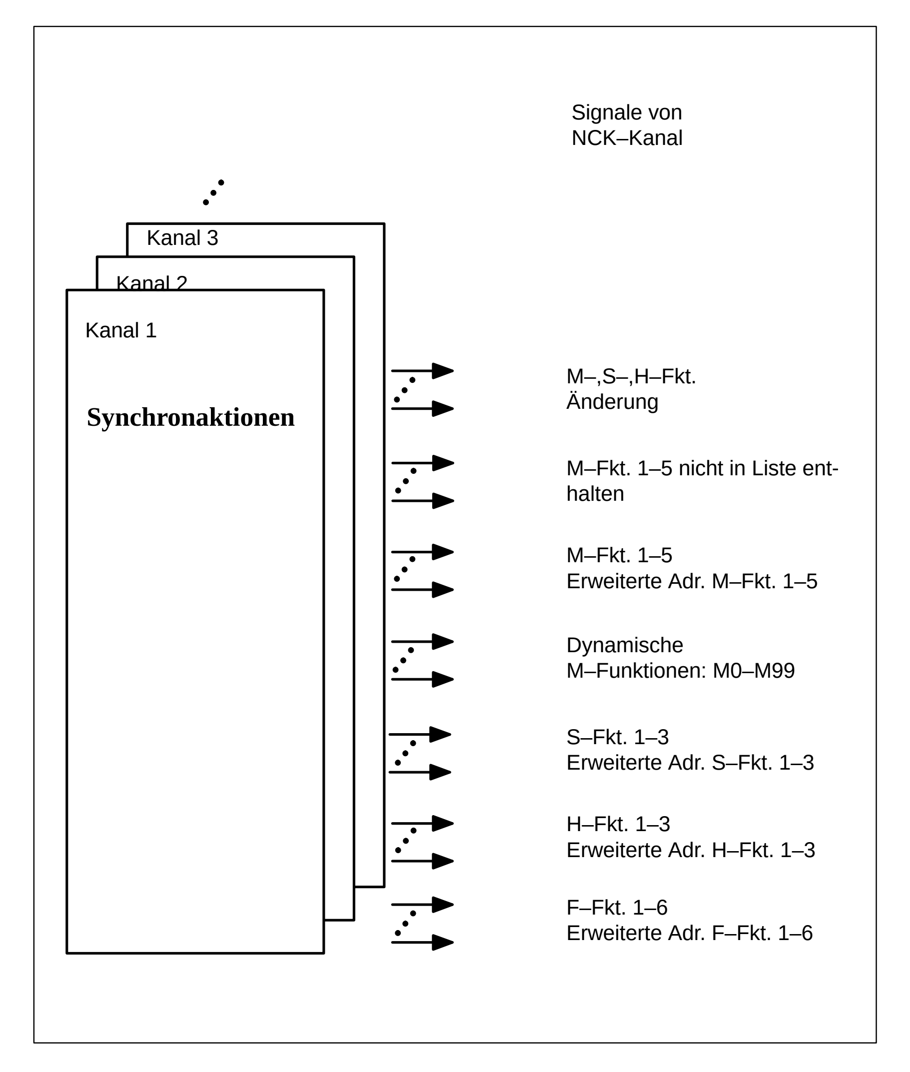



Bild 5-1 PLC–Nahtstellensignale für Synchronaktionen

Die durch Hilfsfunktionenausgabe aus Synchronaktionen erzeugten Signale entsprechen denen in

**Literatur:** /FB/, H2, Hilfsfunktionsausgabe an PLC
beschriebenen Signalen.

**Signale an Kanal**

Mit den Signalen DB 21 – 30 DBB 300 Bit 0 bis
DB 21 – 30 DBB 307 Bit 7
fordert das PLC–Anwendungsprogramm die Sperrung der zugeordneten Synchronaktionen an. Dabei entspricht:
DBB 300 Bit 0 der ersten modalen Synchronaktion (ID=1/IDS=1) und
DBB 307 Bit 7 der 64. modalen Synchronaktion (ID=64/IDS=64).

---

**Hinweis**

Nur die Instanz (NCK oder PLC), die eine Sperre setzt, kann auch wieder die Sperre aufheben.

---

**Signale von Kanal**

Mit den Signalen DB 21 – 30 DBB 308 Bit 0 bis
DB 21 – 30 DBB 315 Bit 7
zeigt der Kanal dem PLC–Anwendungsprogramm an, welche Synchronaktionen durch PLC gesperrt werden dürfen. Dabei entspricht:
DBB 308 Bit 0 der ersten modalen Synchronaktion (ID=1/IDS=1) und
DBB 315 Bit 7 der 64. modalen Synchronaktion (ID=64/IDS=64).

**Alle Synchronaktionen sperren**

Das globale Signal

DB21–30 DBB1 Bit 2

sperrt **alle** modalen/statischen Synchronaktionen, soweit sie nicht geschützt sind.

**Markierte Synchronaktionen sperren**

DB 21 – 30 DBX280.1
Von den in DB 21 – 30 DBB 308 Bit 0 bis DB 21 – 30 DBB 315 Bit 7 als sperrbar markierten Synchronaktionen sollen die in DB 21 – 30 DBB 300 Bit 0 bis DB 21 – 30 DBB 307 Bit 7 durch gesetzte Bit als zu sperren gekennzeichneten Synchronaktionen gesperrt werden.

**Synchronaktionen gesperrt**

DB 21 – 30 DBX281.1
Die angeforderten Synchronaktionen wurden von NCK als gesperrt bestätigt.

■

# Beispiele 6

## 6.1 Beispiele für Bedingungen in Synchronaktionen

**Bahnabstand vom Satzende**

Axialer Abstand 10 mm oder weniger vom Satzende (Werkstück–Koordinatensystem ):
... WHEN $AC_DTEW <= 10 DO ...
G1 X10 Y20

**Achsabstand vom Bahnende**

... WHEN $AA_DTEW[X]<= 10 DO ...
POS[X]= 10

**Bahnabstand vom Satzanfang**

Bahnweg 20 mm oder mehr nach Satzanfang im Basis–Koordinatensystem:
...WHEN $AC_PLTBB >= 20 DO ...

**Bedingung mit Funktion im Vergleich**

Istwert für Achse y im MKS größer als 10 mal Sinus des Wertes in R10:
... WHEN $AA_IM[y] > 10*SIN(R10) DO ...

**Schrittweise Positionieren**

Jedesmal, wenn der Eingang 1 gesetzt wird, wird Achse um einen Schritt weiter positioniert. Der Eingang muß dann wieder zurückgesetzt werden, damit Neustart möglich ist.
G91
EVERY $A_IN[1]==1 DO POS[X]= 10

**In jedem Interpolationstakt OVR**

Um eine Bahnbewegung gezielt festzuhalten, bis ein erwartetes Signal eintrifft, muß $AC_OVR in jedem Interpolationstakt (Schlüsselwort WHENEVER) auf Null gesetz werden.
WHENEVER $A_IN[1]==0 DO $AC_OVR= 0

**Weitere Möglichkeiten**

Die Liste der in Synchronaktionen lesbaren Systemvariablen in

**Literatur:** /PGA/, Programmieranleitung Arbeitsvorbereitung und in Kapitel 2.3.8.

erschließt die volle Menge der in Bedingungen von Synchronaktionen auswertbaren Größen.

## 6.2 Schreiben und Lesen von SD/MD aus Synchronaktionen

**Zustellung und Pendeln beim Schleifen**

Settingdaten, deren Wert während der Bearbeitung unverändert bleiben, werden wie im Teileprogramm mit ihrem gewöhnlichen Namen angesprochen.

**Beispiel**: Pendeln aus Synchronaktionen

| NC–Sprache | Kommentar |
| --- | --- |
| N610 ID=1 WHENEVER $AA_IM[Z]>**$SA_OSCILL_REVERSE_POS1[Z]** DO $AC_MARKER[1]=0 | ;Immer wenn die aktuelle Position der Pendelachse<br>;im Maschinenkoordinatensystem<br>;kleiner als der Beginn des Umkehrbereichs 2 ist,<br>; dann setze den axiale Override der<br>; Zustellachse auf 0 |
| N620 ID=2 WHENEVER $AA_IM[Z]<**$SA_OSCILL_REVERSE_POS2[Z]**–6 DO $AA_OVR[X]=0 $AC_MARKER[0]=0 | ;Immer wenn die aktuelle Position der Pendelachse<br>;im Maschinenkoordinatensystem<br>;gleich der Umkehrposition 1 ist,<br>; dann setze den axialen Override der<br>; Pendelachse auf 0<br>; und setze den axialen Override der<br>; Zustellachse auf 100% (damit wird die<br>; vorhergehende Synchronaktion<br>; aufgehoben!) |
| N630 ID=3 WHENEVER $AA_IM[Z]==**$SA_OSCILL_REVERSE_POS1[Z]** DO $AA_OVR[Z]=0 $AA_OVR[X]=100 | ;Immer wenn der Restweg der Teilzustellung<br>;gleich 0 ist,<br>; dann setze den axialen Override der Pendel-<br>; achse auf 100% (damit wird die vorher-<br>; gehende Synchronaktion aufgehoben!) |
| N640 ID=4 WHENEVER $AA_DTEPW[X]==0<br>DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1<br>N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0<br>N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0 | ;Wenn die aktuelle Position der Pendelachse im<br>;Werkstückkoordinatensystem<br>;gleich der Umkehrposition 1 ist,<br>; dann setze den axialen Override der<br>; Pendelachse auf 100%<br>; und setze den axialen Override der<br>; Zustellachse auf 0 (damit wird die<br>; zweite Synchronaktion einmalig<br>; aufgehoben!) |

N670 ID=7 WHEN $AA_IM[Z]==**$SA_OSCILL_REVERSE_POS1**[Z]
DO $AA_OVR[Z]=100 $AA_OVR[X]=0

Settingdaten, deren Wert sich während der Bearbeitung ändert (z.B. per Bedienung oder Synchronaktion) müssen mit **$$S**... programmiert werden:

**Beispiel**: Pendeln aus Synchronaktionen mit Änderung der Pendelposition von der Bedienoberfläche

N610 ID=1 WHENEVER $AA_IM[Z]>**$$SA_OSCILL_REVERSE_POS1[Z]** DO $AC_MARKER[1]=0

```
;Immer wenn die aktuelle Position der Pendelachse
;im Maschinenkoordinatensystem
;kleiner   als der Beginn des Umkehrbereichs 2 ist,
; dann     setze den axiale Override der
;          Zustellachse auf 0
```

N620 ID=2 WHENEVER $AA_IM[Z]<**$$SA_OSCILL_REVERSE_POS2[Z]**–6
DO $AA_OVR[X]=0 $AC_MARKER[0]=0

```
;Immer wenn die aktuelle Position der Pendelachse
;im Maschinenkoordinatensystem
;gleich    der Umkehrposition 1 ist,
; dann     setze den axialen Override der
;          Pendelachse auf 0
; und      setze den axialen Override der
;          Zustellachse auf 100% (damit wird die
;          vorhergehende Synchronaktion
;          aufgehoben)
```

N630 ID=3 WHENEVER $AA_IM[Z]==**$$SA_OSCILL_REVERSE_POS1[Z]**
DO $AA_OVR[Z]=0 $AA_OVR[X]=100

```
;Immer wenn der Restweg der Teilzustellung
; gleich   0 ist,
; dann     setze den axialen Override der
;          Pendelachse auf 100% (damit wird die
;          vorhergehende Synchronaktion
;          aufgehoben)
```

N640 ID=4 WHENEVER $AA_DTEPW[X]==0
DO $AA_OVR[Z]=100 $AC_MARKER[0]=1 $AC_MARKER[1]=1
N650 ID=5 WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0
N660 ID=6 WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0

```
;Wenn      die aktuelle Position der Pendelachse im
;          Werkstückkoordinatensystem
;gleich    der Umkehrposition 1 ist,
; dann     setze den axialen Override der
;          Pendelachse auf 100%
; und      setze den axialen Override der
;          Zustellachse auf 0 (damit wird die
;          zweite Synchronaktion einmalig
;          aufgehoben)
```

N670 ID=7 WHEN $AA_IM[Z]==**$$SA_OSCILL_REVERSE_POS1[Z]**
DO $AA_OVR[Z]=100 $AA_OVR[X]=0

## 6.3 Beispiele zur AC–Regelung

**Allgemeines Vorgehen**

Die folgenden Beispiele benutzen die Polynomauswertefunktion SYNFCT().

1. Darstellung des Zusammenhangs zwischen Eingangswert und Ausgangswert (jeweils Echtzeitvariablen)
2. Definition dieses Zusammenhanges als Polynom mit Begrenzungen
3. bei Positionsoffset: Setzen der MD und SD
   – MD 36750: $AA_OFF_MODE
   – SA 43350: $SA_AA_OFF_LIMIT (optional)
4. Aktivierung der Regelung in einer Synchronaktion

### 6.3.1 Abstandsregelung mit variabler Obergrenze

**Beispiel für Polynom mit dyn. Obergrenze**

Für eine Abstandsregelung wird die Obergrenze des Ausgangs ($AA_OFF, Überlagerungswert in Achse V) in Abhängigkeit vom Spindeloverride (Analogeingang 1) verändert. Die obere Begrenzung für das Polynom 1 wird dynamisch in Abhängigkeit von Analogeingang 2 verändert.
Es wird das Polynom 1 direkt über die Systemvariablen definiert:

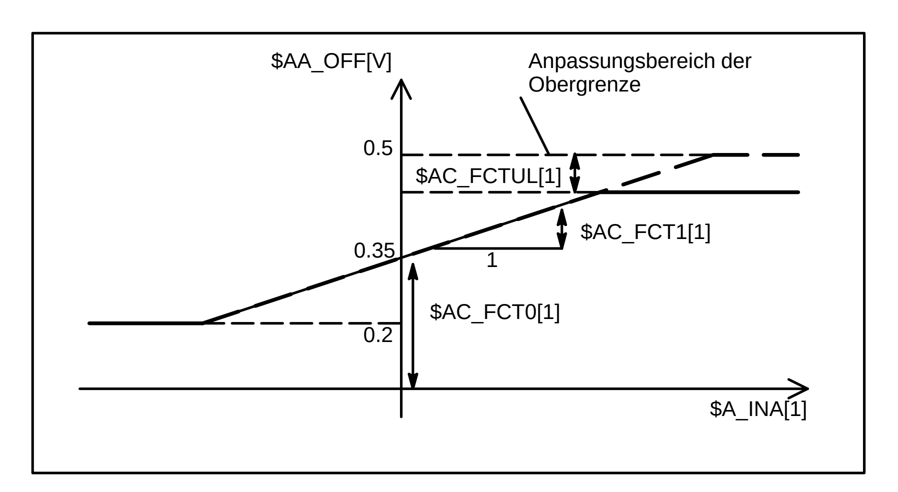



Bild 6-1 Abstandsregelung mit variabler Obergrenze

```
$AC_FCTLL[1]=0.2                    ; Untere Begrenzung
$AC_FCTUL[1]=0.5                    ; Anf. Wert obere Begrenzung
$AC_FCT0[1]=0.35                    ; Nulldurchgang a0
$AC_FCT1[1]=1.5 EX–5                ; Steigung a1
STOPRE                              ; Siehe folgender Hinweis
...
ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1+0.35    ;obere Begrenzung
                                    ; dynamisch anpassen über
                                    ; Analogeingang 2, keine Bedingung
ID=2 DO SYNFCT(1, $AA_OFF[V], $A_INA[1])
                                    ; Abstandsregelung durch Überlagerung
                                    ; keine Bedingung
...
```

**Hinweis**

Bei Verwendung von Systemvariablen im Teileprogramm muß durch Programmierungvon STOPRE für satzsynchrones Schreiben gesorgt werden. Gleichwertig zur obigen Notation zur Polynomdefinition ist:
FCTDEF(1,0.2, 0.5, 0.35, 1.5EX–5).

## 6.3.2 Regelung des Vorschubs

**Beispiel für AC–Regelung mit einer analogen Eingangs–Spannung**

Es soll eine Prozeßgröße (gemessen über $A_INA[1] ) durch Korrektur des Bahn– (oder axialen) Vorschubs additiv beeinflußt auf 2V geregelt werden. Die Vorschubkorrektur soll in den Grenzen ±100 [mm/min] erfolgen.

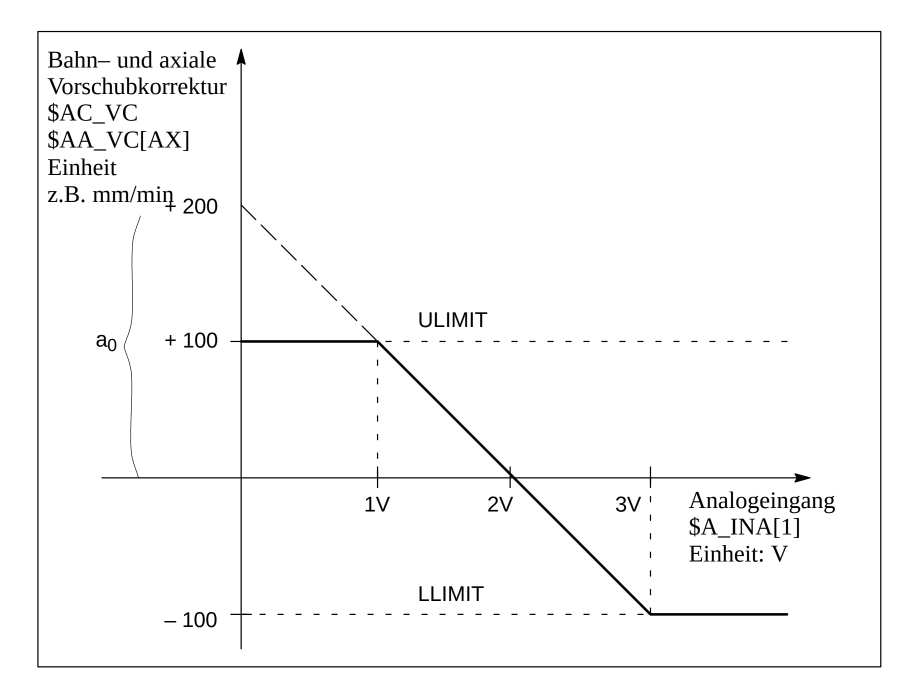



Bild 6-2 Diagramm für AC–Regelung

Bestimmung der Koeffizienten:

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -\frac{100\ \text{mm}}{1\ \text{min} \cdot 1\ \text{V}}$

$a_1 = -100 \Rightarrow$ Regelkonstante, Steigung

$a_0 = -(-100) \cdot 2 = 200$

$a_2 = 0$ (kein quadratisches Glied)

$a_3 = 0$ (kein kubisches Glied)

upper limit = 100

lower limit = –100

```
FCTDEF(      Polynom–Nr,
             LLIMIT,
             ULIMIT,
             a0,            ; y für x = 0
             a1,            ; Steigung
             a2,            ; quadratisches Glied
             a3 )           ; kubisches Glied
```

Mit den oben bestimmten Werten lautet die Polynomdefinition:

FCTDEF(1, –100, 100, 200, –100, 0, 0)

Mit folgenden Synchronaktionen kann die AC–Regelung eingeschaltet werden: für den Achsvorschub:

ID = 1 DO SYNFCT(1, $AA_VC[X], $A_INA[1])

oder für den Bahnvorschub:

ID = 2 DO SYNFCT(1, $AC_VC, $A_INA[1])

### 6.3.3 Geschwindigkeit in Abhängigkeit vom normierten Bahnweg regeln

**Multiplikative Anpassung**

Als Eingangsgröße wird der normierte Bahnweg benutzt: $AC_PATHN.
0: am Satzanfang
1: am Satzende
Die Änderungsgröße $AC_OVR soll in Abhängigkeit von $AC_PATHN nach einem Polynom 3. Ordnung geregelt werden. Der Override soll während der Bewegung von 100 auf 1% reduziert werden.

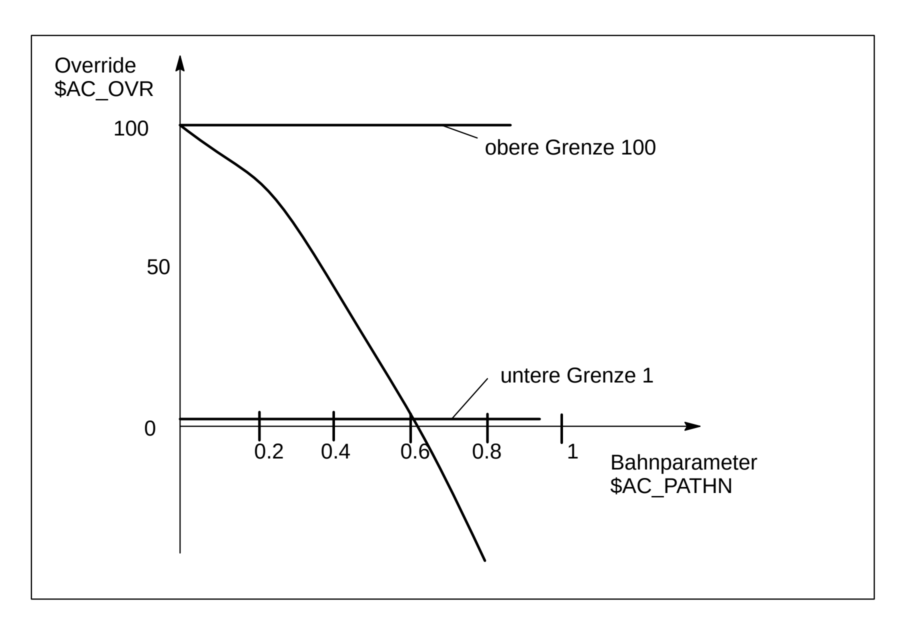



Bild 6-3 Geschwindigkeit kontinuierlich regeln

Polynom 2:
Untere Grenze: 1
Obere Grenze: 100
$a_0$: 100
$a_1$: –100
$a_2$: –100
$a_3$: nicht benutzt

Mit diesen Werten lautet die Polynomdefinition:

```
FCTDEF(2, 1, 100, 100, –100, –100)
; Aktivierung des Veränderlichen Override abhängig vom Bahnweg:

ID= 1 DO SYNFCT(2, $AC_OVR, $AC_PATHN)
G01 X100 Y100 F1000
```

## 6.4 Überwachung eines Sicherheitsabstandes zwischen zwei Achsen

**Aufgabe**

Die Achsen X1 und X2 bedienen zwei unabhängig gesteuerte Transporteinrichtung zum Be– und Entladen von Werkstücken.
Um Kollisionen zu vermeiden, muß zwischen beiden ein Sicherheitsabstand eingehalten werden.
Wird der Sicherheitsabstand unterschritten, so wird die Achse X2 abgebremst.
Die Verriegelung gilt so lange, bis Achse X1 den Sicherheitsbereich wieder verlassen hat.
Bewegt sich Achse X1 weiter auf Achse X2 zu und unterschreitet eine engere Sicherheitsschranke, so fährt sie in eine sichere Position.

| NC–Sprache | Kommentar |
|---|---|
| ID=1 WHENEVER $AA_IM[X2] – $AA_IM[X1] < 30 DO $AA_OVR[X2]=0 | ; Sicherheitsschranke |
| ID=2 EVERY $AA_IM[X2] – $AA_IM[X1] < 15 DO POS[X1]=0 | ; Sichere Position |

## 6.5 Ausführungszeiten in R–Parameter ablegen

**Aufgabe**

Lege ab R–Parameter 10 die Ausführungszeit für die Teileprogrammsätze ab.

| Programm | Kommentar |
|---|---|
| | ; **Ohne** symbolische Programmierung sieht das<br>; Beispiel so aus: |
| IDS=1 EVERY $AC_TIMEC==0 DO $AC_MARKER[0] = $AC_MARKER[0] + 1 | |
| | ; bei Satzwechsel R–Parameter–Zeiger weiterstellen |
| IDS=2 DO $R[10+$AC_MARKER[0]] = $AC_TIME | |
| | ; Schreibe jeweils die aktuelle Zeit vom Satzanfang in<br>; R–Parameter |
| | ; **Mit** symbolischer Programmierung sieht das<br>; Beispiel so aus: |
| DEFINE INDEX AS $AC_MARKER[0] | ; Vereinbarungen für symbolische Programmierung |
| IDS=1 EVERY $AC_TIMEC==0 DO INDEX = INDEX + 1 | ; bei Satzwechsel R–Parameter–Zeiger weiterstellen |
| IDS=2 DO $R[10+INDEX] = $AC_TIME | |
| | ; Schreibe jeweils die aktuelle Zeit vom Satzanfang in<br>; R–Parameter |

## 6.6 ”Einmitten” mit kontinuierlichem Messen

**Einführung**

Es werden nacheinander die Zahnradlücken vermessen. Aus der Summe der Lücken und der Zähnezahl wird das Lückenmaß ermittelt. Die gesuchte Mittenposition für die Weiterbearbeitung ist die Position des ersten Meßpunktes plus 1/2 der durchschnittlichen Lückengröße. Beim Messen wird die Drehzahl so gewählt, daß pro Interpolationstakt ein Meßwert sicher erfaßt werden kann.

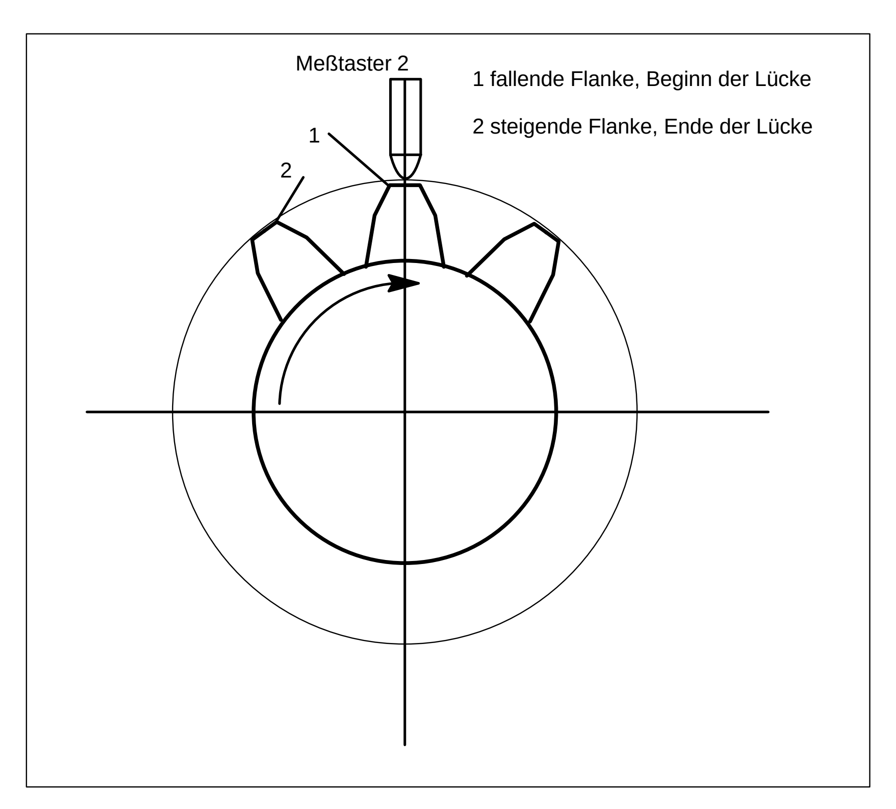



Bild 6-4 Schemabild zum Messen der Zahnradlücken

```
%_N_MEAC_MITTEN_MPF

;Messen mit der Rundachse B (BACH) mit Anzeige der Differenz zwischen den Messwerten


;****** Lokale Anwender– Variablen definieren ***
N1 DEF INT ZAEHNEZAHL                ; Eingabe Anzahl Zahnradzähne
N5 DEF REAL HYS_POS_FLANKE           ; Hysterrese positive Flanke Taster
N6 DEF REAL HYS_NEG_FLANKE           ; Hysterrese negative Flanke Taster

;********** Kurznamen für Synchronaktionsmerker definieren ***********
define M_ZAEHNE as $AC_MARKER[1]     ; ID Merker zum Rechnen: neg/pos Flanke je Zahn
define Z_MW as $AC_MARKER[2]         ; ID Zähler MW FIFO auslesen
define Z_RW as $AC_MARKER[3]         ; ID Zähler MW Rechnen Zahnlücken
```

```
;****** Eingabewerte für ZAHNRADMESSEN *******
N50 ZAEHNEZAHL=26                        ; Eingabe Anzahl zu messende Zahnradzähne
N70 HYS_POS_FLANKE = 0.160               ; Hysterrese positive Flanke Taster
N80 HYS_NEG_FLANKE = 0.140               ; Hysterrese negative Flanke Taster

anfang:                                  ;******* Variablen zuweisen ************************
R1=0                                     ; ID2 Rechenergebnis Lückenmass
R2=0                                     ; ID2 Rechenergebnis Addition aller Lücken
R3=0                                     ; Inhalt des zuerst eingelesenen Elements
R4=0                                     ; R4 Entspricht einem Zahnabstand
R5=0                                     ; Lückenpositon errechnet, Endergebnis
R6=1                                     ; ID 3 BACH mit MOV einschalten
R7=1                                     ; ID 5 MEAC einschalten
M_ZAEHNE=ZAEHNEZAHL*2                    ; ID rechnen neg./pos. Flanke je Zahn
Z_MW=0                                   ; ID Zähler MW FIFO auslesen bis Zähnezahl
Z_RW=2                                   ; ID Zähler Rechnen Differenz Zahnlücke
R13=HYS_POS_FLANKE                       ; Hysterese in Rechenregister
R14=HYS_NEG_FLANKE                       ; Hysterese in Rechenregister

;******* Achse fahren, messen, rechnen **********
N100 MEAC[BACH]=(0)                      ; Meßauftrag rücksetzen
;Rücksetzen der FIFO1[4] Variablen und Sicherstellen eines definierten Messtrace
N105 $AC_FIFO1[4]=0                      ; FIFO1 rücksetzen
STOPRE

; ******* FIFO auslesen bis Zähnezahl erreicht*****
; wenn FIFO1 nicht leer und noch nicht alle Zähne gemessen, Meßwert aus FIFO–Variable in
; Synchronaktionsparameter umspeichern und Zähler Meßwerte erhöhen

ID=1 WHENEVER ($AC_FIFO1[4]>=1) AND (Z_MW<M_ZAEHNE)
      DO $AC_PARAM[0+Z_MW]=$AC_FIFO1[0] Z_MW=Z_MW+1


; wenn 2 Meßwerte vorhanden sind, anfangen zu rechnen, NUR Lückenmaß rechnen und Lückensumme
; Rechenwertzähler um 2 erhöhen

ID=2 WHENEVER (Z_MW>=Z_RW) AND (Z_RW<M_ZAEHNE)
      DO $R1=($AC_PARAM[–1+Z_RW]–$R13)–($AC_PARAM[–2+Z_RW]–$R14) Z_RW=Z_RW+2 $R2=$R2+$R1

; ****** Einschalten der Achse BACH als endlos drehende Rundachse mit MOV *********
WAITP(BACH)
ID=3 EVERY $R6==1 DO  MOV[BACH]=1  FA[BACH]=1000                ; einschalten
ID=4 EVERY $R6==0 und ($AA_STAT[BACH]==1) DO MOV[BACH]=0        ; ausschalten

; Messen nacheinader, Ablegen in FIFO 1, MT2 neg, MT2 pos Flanke
; gemessen wird der Abstand zwischen 2 Zähnen fallende Flanke–...–steigende Flanke ,Taster 2
N310 ID=5 WHEN $R7==1 DO MEAC[BACH]=(2, 1, –2, 2)
N320 ID=6 WHEN (Z_MW>=M_ZAEHNE) DO MEAC[BACH]=(0)               ; Messung abbrechen
M00
STOPRE


; ******* FIFO Werte holen und abspeichern ***
N400 R3=$AC_PARAM[0]                     ; Inhalt des zuerst eingelesenen Elements
                                         ; Rücksetzen der FIFO1[4] Variablen und Sicherstellen
                                         ; eines definierten Meßtrace für nächsten Meßauftrag

N500 $AC_FIFO1[4]=0


; ******* Differenz zwischen den einzelnen Zaehnen rechen
N510 R4=R2/(ZAEHNEZAHL)/1000             ; R4 Entspricht einem durchschnittlichen Zahnabstand
                                         ; Division ”/1000” entfällt in späteren SW–Ständen
```

```
; ******** Mittenposition berechnen **********
N520 R3=R3/1000 ; Erste Meßpostion auf Grad umgerechnet
N530 R3=R3 MOD 360 ;ersten Meßpunkt modulo
N540 R5=(R3–R14)+(R4/2) ; Lückenpositon rechnen
M00
stopre
R6=0                              ; Achsdrehung von BACH ausschalten
gotob anfang
M30
```

# 6.7 Achskopplungen über Synchronaktionen

## 6.7.1 Einkoppeln auf Leitachse

**Aufgabenstellung**

Über Polynomsegmente wird eine zyklische Kurventabelle definiert. Gesteuert über Rechenvariablen wird die Bewegung der Leitachse und der Koppelvorgang zwischen Leitachse und abhängiger Achse ein–/ausgeschaltet.

```
%_N_KOP_SINUS_MPF


N5 R1=1                    ; ID 1, 2 ein–/ausschalten der Kopplung: LEADON (CACB, BACH)
N6 R2=1                    ; ID 3, 4  Leitachse bewegen ein–/aus: MOV BACH
N7 R5=36000                ; BACH Vorschub/min
N8 STOPRE

;**** Periodische Tabelle Nr. 4 durch Polynomsegmente definieren ****
N10 CTABDEF (YGEO,XGEO,4,1)
N16 G1 F1200  XGEO=0.000 YGEO=0.000     ; Grundstellungen anfahren
N17  POLY PO[XGEO]=(79.944,3.420,0.210) PO[YGEO]=(24.634,0.871,–9.670)
N18  PO[XGEO]=(116.059,0.749,–0.656) PO[YGEO]=(22.429,–5.201,0.345)
N19  PO[XGEO]=(243.941,–17.234,11.489) PO[YGEO]=(–22.429,–58.844,39.229)
N20  PO[XGEO]=(280.056,1.220,–0.656) PO[YGEO]=(–24.634,4.165,0.345)
N21  PO[XGEO]=(360.000,–4.050,0.210) PO[YGEO]=(0.000,28.139,–9.670)
N22 CTABEND    ; **** Ende  der  Tabellendefinition*****


; Achse Leitachse und gekoppelte Achse im Eilgang in Grundstellung fahren
N80 G0 BACH=0 CACH=0            ; Kanalachsnamen
N50 LEADOF(CACH,BACH)           ; ggf. bestehende Kopplung AUS


N235 ;******* Einschalten der Koppel–Bewegung für die Achse CACH *******
N240 WAITP(CACH)                 ; Achse auf Kanal synchronisieren
N245 ID=1 EVERY $R1==1 DO LEADON(CACH, BACH, 4)       ; Über Tabelle 4 einkoppeln
N250 ID=2 EVERY $R1==0 DO LEADOF(CACH, BACH)          ; Kopplung ausschalten

N265 WAITP(BACH)

N270 ID=3 EVERY $R2==1 DO MOV[BACH]=1  FA[BACH]=R5   ; Leitachse mit Vorschub in R5 endlos drehen
N275 ID=4 EVERY $R2==0 DO MOV[BACH]=0                 ; Leitachse anhalten

N280 M00
N285 STOPRE
N290 R1=0                              ; Ausschalten Koppelbedingung
N295 R2=0                              ; Ausschalten Bedingung für Leitachse drehen
N300 R5=180                 ; Neuer Vorschub für  BACH
N305 M30
```

## 6.7.2 Unrundschleifen über Leitwertkopplung

**Aufgabenstellung**

Ein unrundes Werkstück, das sich auf der Achse CACH dreht, soll durch Schleifen bearbeitet werden. Der Abstand der Schleifscheibe vom Werkstück wird über die Achse XACH gesteuert. Er hängt von der Drehlage des Werkstükkes ab. Der Zusammenhang zwischen den Drehlagen und zugeordneten Bewegungen ist durch Kurventabelle 2 definiert. Das Werkstück soll sich mit Geschwindigkeiten bewegen, die von der Werkstückkontur gemäß Kurventabelle 1 abhängen.

**Lösung**

CACH wird zu Leitachse einer Leitwertkopplung. Sie wirk:
– über Tabelle 2 auf die Ausgleichsbewegung der Achse XACH
– über Tabelle 1 auf die ”Softwareachse” CASW.

Der Achsoverride der Achse CACH bestimmt sich aus den Istwerten der Achse CASW. Damit ist die geforderte konturabhängige Geschwindigkeit der Achse CACH realisiert.

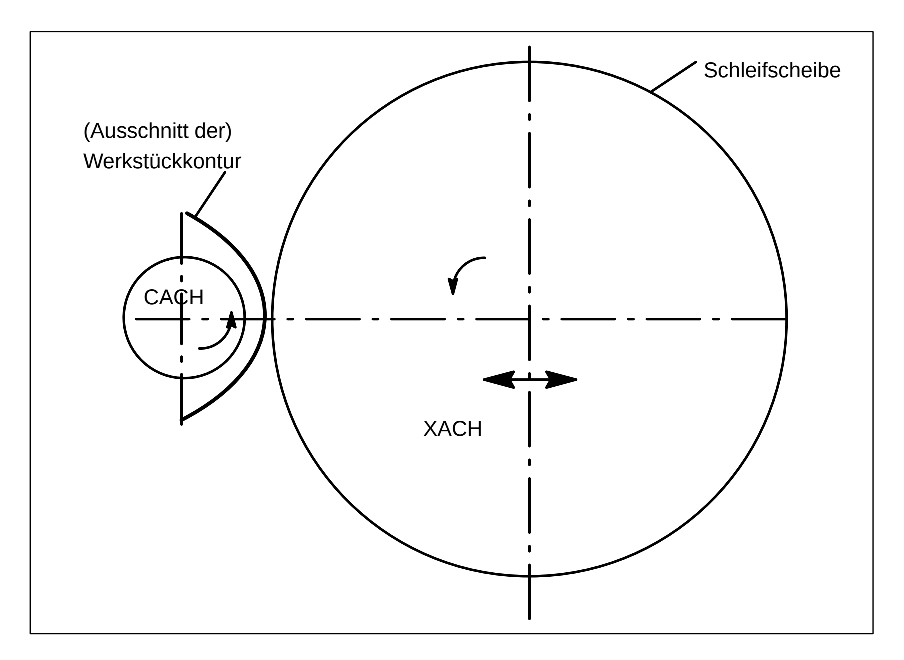



Bild 6-5 Schema Schleifen einer Unrund–Kontur

```
%_N_CURV_TABS_SPF
PROC CURV_TABS
N160   ; *************** Tabelle 1 Override definieren ******
N165 CTABDEF(CASW,CACH,1,1)            ; Tabelle 1 periodisch
N170 CACH=0 CASW=10
N175 CACH=90 CASW=10
N180 CACH=180 CASW=100
N185 CACH=350 CASW=10
N190 CACH=359.999 CASW=10
N195 CTABEND
```

```
N160    ; **** Tabelle 2 Linieare Ausgleichbewegung der XACH definieren ******
CTABDEF(YGEO,XGEO,2,1)                  ; Tabelle 2 periodisch
N16 XGEO=0.000 YGEO=0.000
N16 XGEO=0.001 YGEO=0.000
N17  POLY PO[XGEO]=(116.000,0.024,0.012) PO[YGEO]=(4.251,0.067,–0.828)
N18  PO[XGEO]=(244.000,0.072,–0.048) PO[YGEO]=(4.251,–2.937)
N19  PO[XGEO]=(359.999,–0.060,0.012) PO[YGEO]=(0.000,–2.415,0.828)
N16 XGEO=360.000 YGEO=0.000
N20 CTABEND

M17


%_N_UNRUND_MPF
; Kopplelverbund für eine Unrundbearbeitung

; XACH ist die Zustellachse der Schleifscheibe
; CACH ist die Werkstückachse als Rundachse und Leitwertachse
; Anwendung: Unrunde Kontur schleifen
; Tabelle 1 bildet den Override für Achse CACH als Funktion der Position von CACH ab
; Überlagerung der XGEO Achse mit Handrad Zustellung für Ankratzen

N100 DRFOF                        ; Handradüberlagerung abwählen
N200 MSG("DRF anwaehlen,  (Handrad 1 aktiv)  und Anwahl INKREMENT.== Handradueberlagerung AKTIV")
N300 M00
N500 MSG()                        ; Meldung rücksetzen
N600 R2=1                         ; LEADON Tabelle 2,  Einschalten mit ID=3/4  CACH auf XACH
N700 R3=1                         ; LEADON Tabelle 1 , Einschalten mit ID=5/6  CACH auf CASW, Override
N800 R4=1                         ; Endlos drehende Rundachse CACH, Start mit ID=7/8
N900 R5=36000                     ; FA[CACH]  Endlos drehende Rundachse Drehzahl

N1100 STOPRE
N1200                             ; *********  Achsen und Leitachse auf FA einstellen *******
                                  ; Achse Leitachse und Folgeachse in Grundstellung fahren
N1300 G0 XGEO=0 CASW=10 CACH=0
N1400 LEADOF(XACH,CACH)           ; Kopplung AUS XACH Ausgleichsbewegung
N1500 LEADOF(CASW,CACH)           ; Kopplung AUS CASW Overridetabelle

N1600 CURV_TABS                   ; Unterprogamm mit der Definition der Tabellen


N1700                             ; ******* Einschalter der LEADON Ausgleichbewegung XACH *******
N1800 WAITP(XGEO)                 ; Achse auf Kanal synchronisieren
N1900 ID=3 EVERY $R2==1 DO LEADON(XACH,CACH,2)
N2000 ID=4 EVERY $R2==0 DO LEADOF(XACH,CACH)

N2100                             ; ************ Einschalter der LEADON CASW Overridetabelle ****
N2200 WAITP(CASW)
N2300 ID=5 EVERY $R3==1 DO LEADON(CASW,CACH,1)       ; CTAB Kopplung  EIN Leitachse CACH
N2400 ID=6 EVERY $R3==0 DO LEADOF(CASW,CACH)         ; CTAB Kopplung AUS Leitachse CACH


N2500                             ; ** Override der CACH von Position CASW mit ID 10 beeinflussen *
N2700 ID=11 DO $$AA_OVR[CACH]=$AA_IM[CASW]           ; "Achsposition" CASW auf OVR CACH zuweisen

N2900 WAITP(CACH)

N3000 ID=7 EVERY $R4==1 DO MOV[CACH]=1 FA[CACH]=R5   ; Als endlos drehende Rundachse starten
N3100 ID=8 EVERY $R4==0 DO MOV[CACH]=0               ; Als endlos drehende Rundachse anhalten

N3200 STOPRE
N3300 R90=$AA_COUP_ACT[CASW]                         ; Zustand der Kopplung für CASW zum Prüfen
N3400 MSG("Overridetabelle CASW eingeschalten mit LEADON "<<R90<<", weiter ENDE mit NC–START")
```

```
N3500 M00                  ; ********** NC HALT **************
N3600 MSG()
N3700 STOPRE               ; Vorlaufstop
N3800 R1=0                 ; Stop mit ID=2 CASW Achse als endlos drehende Rundachse
N3900 R2=0                 ; LEADOF mit ID=6 FA XACH und Leitachse CACH
N4000 R3=0                 ; LEADOF TAB1 CASW mit ID=7/8  CACH auf CASW Overridetabelle
N4100 R4=0                 ; Achse als endlos drehende Rundachse anhalten,  ID=4 CACH
N4200 M30
```

**Ausbaumöglichkeiten**

Das obige Beispiel läßt sich in folgende Punkten ausbauen:

- Einführung einer Z–Achse, um Schleifscheibe oder Werkstück von einem Unrund zum nächsten auf der gleichen Welle zu bewegen (Nockenwelle).
- Tabellenumschaltungen, wenn die Nocken z.B. für Einlaß und Auslaß verschiedene Konturen haben.
  ID = ... <Bedingung> DO LEADOF(XACH, CACH) LEADON(XACH, CACH, <neue Tabellennummer>)
- Abrichten der Schleifscheibe über online Werkzeugkorrektur gem. 2.4.7.

## 6.7.3 Fliegendes Trennen

**Aufgabenstellung**

Ein Strangmaterial, das sich stetig durch einen Arbeitsbereich einer Trennvorrichtung bewegt, soll in gleichlange Stücke zerteilt werden.

X–Achse: Achse in der sich das Strangmaterial bewegt. WKS
X1–Achse: Maschinenachse des Strangmaterials, MKS
Y–Achse: Achse in der die Trennvorrichtung mit dem Strangmaterial ”mitfährt”

Es wird angenommen, daß die Zustellung des Trennwerkzeuges und seine Steuerung durch PLC kontrolliert wird. Zur Feststellung der Synchronität zwischen Strangmaterial und Trennwerkzeug können die Signale der PLC–Nahtstelle ausgewertet werden.

**Aktionen**

Kopplung einschalten, LEADON
Kopplung ausschalten, LEADOF
Istwertsetzen, PRESETON

**NC–Programm** **Kommentar**

```
%_N_SCHERE1_MPF
;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD
N100 R3=1500                                ; Länge eines abzutrennenden Teiles
N200 R2=100000      R13=R2/300
N300 R4=100000
N400 R6=30                                  ; Startposition  Y Achse
N500 R1=1                                   ; Startbedingung für Bandachse
N600 LEADOF(Y,X)                            ; löschen einer evtl. bestehenden Kopplg.
N700 CTABDEF(Y,X,1,0)                       ; Tabellendefinition
N800 X=30 Y=30                              ; Wertepaare
N900 X=R13 Y=R13
N1000 X=2*R13 Y=30
N1100 CTABEND                               ; Ende der Tabelledefinition
N1200 PRESETON(X1,0)                        ; PRESET zu Beginn
N1300 Y=R6 G0                               ; Startpos. Y Achse

                                            ; Achse ist Linear
N1400 ID=1 EVERY $AA_IW[X]>$R3 DO PRESETON(X1,0)        ; PRESET nach Länge R3, PRESTON darf nur mit
                                                        ; WHEN und EVERY erfolgen
                                            ; neuer Beginn nach Abtrennen
N1500 WAITP(Y)

N1800 ID=6 EVERY $AA_IM[X]<10  DO LEADON(Y,X,1)         ; Y über Tabelle 1 an X ankoppeln bei X < 10
N1900 ID=10 EVERY  $AA_IM[X]>$R3–30  DO LEADOF(Y,X)     ; > 30 vor gefahrener Trennlänge abkoppeln
N2000 WAITP(X)
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1  FA[X]=$R4           ; Strangachse stetig in Bewegung setzen
N2200 M30
```

## 6.8 Technologiezyklen Spindel Positionieren

**Anwendung**

Im Zusammenwirken mit dem PLC–Programm soll die Spindel, die einen Werkzeugwechsel antreibt:

– in eine Ausgangsstellung positioniert werden

– auf einen bestimmten Wert positioniert werden, auf dem sich das einzuwechselnde Werkzeug befindet

Vergl. 2.4.12, 2.6.1.

**Koordinierung**

Die Koordinierung zwischen PLC und NCK erfolgt über die ab SW–Stand 4 verfügbaren gemeinsamen Daten (siehe: 2.3.8)

– $A_DBB[0] 1 Grundposition einnehmen

– $A_DBB[1] 1 Zielposition einnehmen

– $A_DBW[1] zu positionierender Wert + / – , PLC berechnet den kürzesten Weg.

**Synchronaktionen** %_N_MAIN_MPF

```
...
IDS=1 EVERY $A_DBB[0]==1 DO  NULL_POS    ; wenn $A_DBB[0] von PLC gesetzt, Grundposition einnehmen
IDS=2 EVERY $A_DBB[1]==1 DO  ZIEL_POS    ; wenn $A_DBB[1] von PLC gesetzt, Spindel auf den in
                                         ; $A_DBW[1] hinterlegten Wert positionieren
...
```

**Technologiezyklus NULL_POS** %_N_NULL_POS_SPF

```
PROC NULL_POS
SPOS=0              ; Antrieb für den Werkzeugwechsel in Grundposition bringen
$A_DBB[0]=0         ; Grundposition in NCK ausgeführt
```

**Technologiezyklus ZIEL_POS** %_N_ZIEL_POS_SPF

```
PROC ZIEL_POS
SPOS=IC($A_DBW[1])  ; Spindel auf den Wert positionieren, der in $A_DBW[1]
                    ; von PLC hinterlegt wurde, Kettenmaß
$A_DBB[1]=0         ; Zielpositionieren in NCK ausgeführt
```

# 6.9 Synchronaktionen im Bereich WZW/BAZ

**Einführung**

Das folgende Bild zeigt den schematischen Ablauf Werkzeugwechselzyklus.

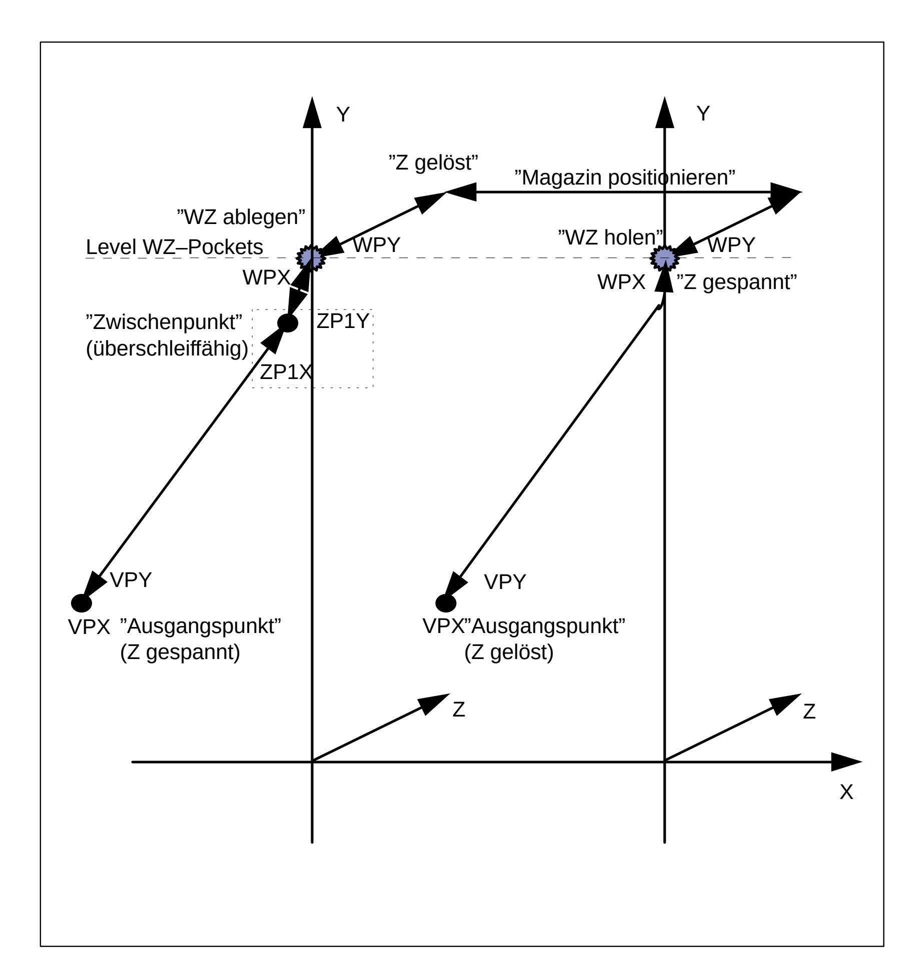



Bild 6-6 Schematischer Ablauf Werkzeugwechselzyklus

## Ablaufdiagramm

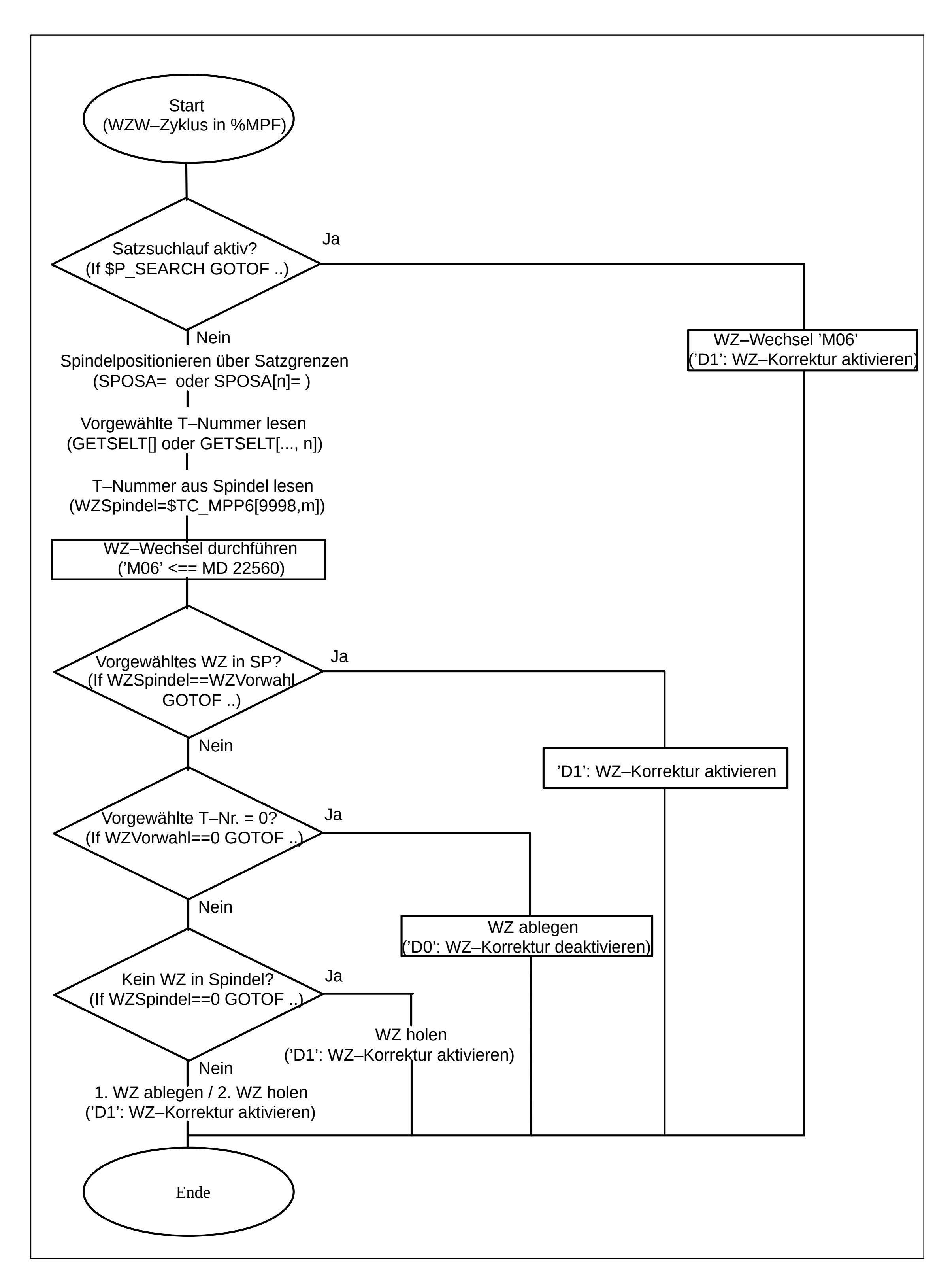



Bild 6-7 Ablaufdiagramm Werkzeugwechselzyklus

**NC–Programm** **Kommentar**

%_N_WZW_SPF

```
;$PATH=/_N_SPF_DIR
N10 DEF INT WZVorwahl,WZSpindel
N15 WHEN $AC_PATHN<10 DO $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[2]=0
N20 ID=3 WHENEVER $A_IN[9]==TRUE DO $AC_MARKER[1]=1        ; Marker auf = 1 wenn MagAchse gefahren
N25 ID=4 WHENEVER $A_IN[10]==TRUE DO $AC_MARKER[2]=1       ; Marker auf = 1 wenn MagAchse gefahren
N30 IF $P_SEARCH GOTOF wzw_vorlauf                         ; Satzvorlauf aktiv ? –>
N35 SPOSA=0 D0
N40 GETSELT(WZVorwahl)                                     ; vorgewählte T–Nr.lesen
N45 WZSpindel=$TC_MPP6[9998,1]                             ; WZ in Spindel lesen
N50 M06
N55 IF WZSpindel==WZVorwahl GOTOF wz_in_spindel  IF WZVorwahl==0 GOTOF ablegen1 IF WZSpindel==0 GO-
TOF holen1

;*****Werkzeug holen und ablegen*****
ablegen1holen1:
N65 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1           ;wenn MagAchse fährt Marker = 1
N70 G01 G40 G53 G64 G90 X=Magazin1VPX Y=Magazin1VPY Z=Magazin1ZGespannt F70000 M=QU(120)
M=QU(123) M=QU(9)
N75 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0                  ; Spindel in Position
N80 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1           ; MagAchse fährt abfragen
N85 WHENEVER $AC_MARKER[1]==0 DO $AC_OVR=0                 ; Override=0 wenn Achse nicht gefahren
N90 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0              ; Override=0 wenn MagAchse nicht in Pos fein
N95 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0                   ; Override=0 wenn Restweg MagAchse > 0
N100 G53 G64 X=Magazin1ZP1X Y=Magazin1ZP1Y F60000
N105 G53 G64 X=Magazin1WPX Y=Magazin1WPY F60000
N110 M20                                                   ; WZ lösen
N115 G53 G64 Z=MR_Magazin1ZGeloest F40000
N120 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1;
N125 WHENEVER $AC_MARKER[2]==0  DO $AC_OVR=0
N130 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0
N135 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0
N140 G53 G64 Z=Magazin1ZGespannt F40000
N145 M18                                                   ; Werkzeug spannen
N150 WHEN $AC_PATHN<10 DO M=QU(150) M=QU(121)              ; Bedingung immer erfüllt
N155 G53 G64 X=Magazin1VPX Y=Magazin1VPY F60000 D1 M17

;*****Werkzeug ablegen*****
ablegen1:
N160 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1
N165 G01 G40 G53 G64 G90 X=Magazin1VPX Y=Magazin1VPY Z=Magazin1ZGespannt F70000 M=QU(120)
M=QU(123) M=QU(9)
N170 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0
N175 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[1]=1
N180 WHENEVER $AC_MARKER[1]==0 DO $AC_OVR=0
N185 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0
N190 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0
N195 G53 G64 X=Magazin1ZP1X Y=Magazin1ZP1Y F60000
N200 G53 G64 X=Magazin1WPX Y=Magazin1WPY F60000
N205 M20                                                   ; Werkzeug lösen
N210 G53 G64 Z=Magazin1ZGeloest F40000
N215 G53 G64 X=Magazin1VPX Y=Magazin1VPY F60000 M=QU(150) M=QU(121) D0 M17

;*****Werkzeug holen*****
holen1:
N220 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1
N225 G01 G40 G53 G64 G90 X=Magazin1VPX Y=Magazin1VPY Z=Magazin1ZGeloest F70000 M=QU(120)
M=QU(123) M=QU(9)
N230 G53 G64 X=Magazin1WPX Y=Magazin1WPY F60000
N235 WHENEVER $AA_STAT[S1]<>4 DO $AC_OVR=0
N240 WHENEVER $AA_VACTM[C2]<>0 DO $AC_MARKER[2]=1
N245 WHENEVER $AC_MARKER[2]==0 DO $AC_OVR=0
N250 WHENEVER $AA_STAT[C2]<>4 DO $AC_OVR=0
```

```
N255 WHENEVER $AA_DTEB[C2]>0 DO $AC_OVR=0
N260 G53 G64 Z=Magazin1ZGespannt F40000
N265 M18                                        ; Werkzeug spannen
N270 G53 G64 X=Magazin1VPX Y=Magazin1VPY F60000 M=QU(150) M=QU(121) D1 M17

;*****Werkzeug in Spindel *****
wz_in_spindel:
N275 M=QU(121) D1 M17

;*****Satzvorlauf*****
wzw_vorlauf:
N280 STOPRE
N285 D0
N290 M06
N295 D1 M17
```

■

## Platz für Notizen

# 7 Datenfelder, Listen

## 7.1 Nahtstellensignale

| DB–Nummer | Bit , Byte | Name | Verweis |
|---|---|---|---|
| **kanalspezifisch** | | | |
| 21–30 | 280.1 | Modale Synchronaktionen gem. DBX 300.0–307.7 sperren | |
| 21–30 | 300.0 – | Modale Synchronaktionen gem. DBX 300.0–307.7 gesperrt, Quittung von NCK | |
| 21–30 | 300.0 – | Modale Synchronaktionen ID oder IDS 1 – | |
| 21–30 | 307.7 | 64 sperren. Anforderung an Kanal der NCK | |
| 21–30 | 308.0 – | Modale Synchronaktionen ID oder IDS 1 – | |
| 21–30 | 315.7 | 64 sperrbar. Mitteilung von NCK. | |

## 7.2 Maschinendaten

| Nummer | Bezeichner | Name | Verweis |
|---|---|---|---|
| **allgemein ($MN_ ... )** | | | |
| 11110 | AUXFU_GROUP_SPEC | Hilfsfunktionsgruppenspezifikation | H2 |
| 11500 | PREVENT_SYNACT_LOCK | Geschützte Synchronaktionen | |
| **kanalspezifisch ($MC_ ... )** | | | |
| 21240 | PREVENT_SYNACT_LOCK_CHAN | Geschützte Synchronaktionen des Kanals | |
| 28250 | MM_NUM_SYNC_ELEMENTS | Anzahl Elemente für Ausdrücke der Synchronaktionen | |
| 28252 | MM_NUM_FCTDEF_ELEMENTS | Anzahl der FCTDEF–Elemente | |
| 28254 | MM_NUM_AC_PARAM | Parameteranzahl $AC_PARAM | |
| 28255 | MM_BUFFERED_AC_PARAM | Speicherort für $AC_PARAM (ab SW 6.3) | |
| 28256 | MM_NUM_AC_MARKER | Merkeranzahl $AC_MARKER | |
| 28257 | MM_BUFFERED_AC_MARKER | Speicherort für $AC_MARKER (ab SW 6.3) | |
| 28258 | MM_NUM_AC_TIMER | Anzahl Zeitvariablen $AC_TIMER | |
| 28260 | NUM_AC_FIFO | Anzahl Variablen $AC_FIFO1, $AC_FIFO2, ... | |
| 28262 | START_AC_FIFO | FIFO–Variablen speichern ab R–Parameter | |
| 28264 | LEN_AC_FIFO | Länge der FIFO–Variablen $AC_FIFO ... | |
| 28266 | MODE_AC_FIFO | Modus der FIFO–Bearbeitung | |

| achsspezifisch ($MA_ ... ) | | | |
|---|---|---|---|
| 30450 | IS_CONCURRENT_POS_AX | Konkurrierende Positionierachse | P2 |
| 32060 | POS_AX_VELO | Löschstellung für Positionierachsgeschwindigkeit | P2 |
| 32070 | CORR_VELO | Achsgeschwindigkeit für Handrad, ext. NPV, cont. Dressing, Abstandsregelung (ab SW3) | H1 |
| 32074 | FRAME_OR_CORRPOS_NOTALLOWED | Wirksamkeit der Frames und Werkzeuglängen-korrektur | |
| 32920 | AC_FILTER_TIME | Filter–Glättungszeitkonstante für Adaptive Con-trol (ab SW2) | |
| | | | |
| 36750 | AA_OFF_MODE | Wirkung der Wertzuweisung für axiale Überlage-rung bei Synchronaktionen (ab SW3) | |
| 37200 | COUPLE_POS_TOL_COARSE | Schwellwert für ”Synchronlauf grob” | S3 |
| 37210 | COUPLE_POS_TOL_FINE | Schwellwert für ”Synchronlauf fein” | S3 |
| **Settingdaten ($SA_ ... )** | | | |
| 43300 | ASSIGN_FEED_PER_REV_SOURCE | Umdrehungsvorschub für Positionierachsen/Spin-deln | V1 |
| 43350 | AA_OFF_LIMIT | Obergrenze des Lorrekturwertes für $AA_OFF Abstandsregelung | |
| 43400 | WORKAREA_PLUS_ENABLE | Arbeitsfeldbegrenzung in pos. Richtung | A3 |

## 7.3 Alarme

Ausführliche Erläuterungen zu den auftretenden Alarmen können der
**Literatur:** /DA/, ”Diagnoseanleitung”
bzw. bei Systemen mit MMC 101/102 der Online–Hilfe entnommen werden.

■

# Literatur

# A

## Allgemeine Dokumentation


**/BU/**

SINUMERIK 840D/840Di/810D/802S, C, D
Bestellunterlage
Katalog NC 60
Bestellnummer: E86060–K4460–A101–A9
Bestellnummer: E86060–K4460–A101–A9 –7600 (englisch)

**/IKPI/**

Katalog IK PI 2000
Industrielle Kommunikation und Feldgeräte
Bestellnummer der gebundenen Ausgabe: E86060–K6710–A101–A9
Bestellnumme der Einzelblattausgaber: E86060–K6710–A100–A9

**/ST7/**

SIMATIC
Speicherprogrammierbare Steuerungen SIMATIC S7
Katalog ST 70
Bestellnummer: E86 060–K4670–A111–A3

**/Z/**

SINUMERIK, SIROTEC, SIMODRIVE
Verbindungstechnik & Systemkomponenten
Katalog NC Z
Bestellnummer: E86060–K4490–A001–A8
Bestellnummer: E86060–K4490–A001–A8 –7600 (englisch)


## Elektronische Dokumentation


**/CD1/**

Das SINUMERIK–System (Ausgabe 11.02)
DOC ON CD
(mit allen SINUMERIK 840D/840Di/810D/802– und SIMODRIVE– Schriften)
Bestellnummer: 6FC5 298–6CA00–0AG3

## Anwender–Dokumentation

**/AUK/** SINUMERIK 840D/810D
**Kurzanleitung Bedienung AutoTurn** (Ausgabe 09.99)
Bestellnummer: 6FC5 298–4AA30–0AP2

**/AUP/** SINUMERIK 840D/810D
**Grafisches Programmiersystem AutoTurn** (Ausgabe 02.02)
Programmieren/Einrichten
Bestellnummer: 6FC5 298–4AA40–0AP3

**/BA/** SINUMERIK 840D/810D
**Bedienungsanleitung MMC** (Ausgabe 10.00)
Bestellnummer: 6FC5 298–6AA00–0AP0

**/BAD/** SINUMERIK 840D/840Di/810D
**Bedienungsanleitung HMI Advanced** (Ausgabe 11.02)
Bestellnummer: 6FC5 298–6AF00–0AP2

**/BEM/** SINUMERIK 840D/810D
**Bedienungsanleitung HMI Embedded** (Ausgabe 11.02)
Bestellnummer: 6FC5 298–6AC00–0AP2

**/BAH/** SINUMERIK 840D/840Di/810D
**Bedienungsanleitung HT 6** (Ausgabe 06.02)
Bestellnummer: 6FC5 298–0AD60–0AP2

**/BAK/** SINUMERIK 840D/840Di/810D
**Kurzanleitung Bedienung** (Ausgabe 02.01)
Bestellnummer: 6FC5 298–6AA10–0AP0

**/BAM/** SINUMERIK 840D/810D
**Bedienen/Programmieren ManualTurn** (Ausgabe 08.02)
Bestellnummer: 6FC5 298–6AD00–0AP0

**/BAS/** SINUMERIK 840D/810D
**Bedienen/Programmieren ShopMill** (Ausgabe 11.02)
Bestellnummer: 6FC5 298–6AD10–0AP1

**/BAT/** SINUMERIK 840D/810D
**Bedienen/Programmieren ShopTurn** (Ausgabe 03.03)
Bestellnummer: 6FC5 298–6AD50–0AP2

| | | |
|---|---|---|
| **/BNM/** | SINUMERIK 840D/840Di/810D<br>**Benutzerhandbuch Meßzyklen**<br>Bestellnummer: 6FC5 298–6AA70–0AP2 | (Ausgabe 11.02) |
| **/CAD/** | SINUMERIK 840D/840Di/810D<br>**Bedienungsanleitung CAD–Reader**<br>Bestellnummer: (ist Bestandteil der Online–Hilfe) | (Ausgabe 03.02) |
| **/DA/** | SINUMERIK 840D/840Di/810D<br>**Diagnoseanleitung**<br>Bestellnummer: 6FC5 298–6AA20–0AP3 | (Ausgabe 11.02) |
| **/KAM/** | SINUMERIK 840D/810D<br>**Kurzanleitung ManualTurn**<br>Bestellnummer: 6FC5 298–5AD40–0AP0 | (Ausgabe 04.01) |
| **/KAS/** | SINUMERIK 840D/810D<br>**Kurzanleitung ShopMill**<br>Bestellnummer: 6FC5 298–5AD30–0AP0 | (Ausgabe 04.01) |
| **/KAT/** | SINUMERIK 840D/810D<br>**Kurzanleitung ShopTurn**<br>Bestellnummer: 6FC5 298–6AF20–0AP0 | (Ausgabe 07.01) |
| **/PG/** | SINUMERIK 840D/840Di/810D<br>**Programmieranleitung Grundlagen**<br>Bestellnummer: 6FC5 298–6AB00–0AP2 | (Ausgabe 11.02) |
| **/PGA/** | SINUMERIK 840D/840Di/810D<br>**Programmieranleitung Arbeitsvorbereitung**<br>Bestellnummer: 6FC5 298–6AB10–0AP2 | (Ausgabe 11.02) |
| **/PGK/** | SINUMERIK 840D/840Di/810D<br>**Kurzanleitung Programmierung**<br>Bestellnummer: 6FC5 298–6AB30–0AP1 | (Ausgabe 02.01) |
| **/PGM/** | SINUMERIK 840D/840Di/810D<br>**Programming Guide ISO Milling**<br>Bestellnummer: 6FC5 298–6AC20–0BP2 | (Edition 11.02) |
| **/PGT/** | SINUMERIK 840D/840Di/810D<br>**Programming Guide ISO Turning**<br>Bestellnummer: 6FC5 298–6AC10–0BP2 | (Edition 11.02) |

**/PGZ/** SINUMERIK 840D/840Di/810D
**Programmieranleitung Zyklen** (Ausgabe 11.02)
Bestellnummer: 6FC5 298–6AB40–0AP2

**/PI /** **PCIN 4.4**
Software zur Datenübertragung an/von MMC–Modul
Bestellnummer: 6FX2 060 4AA00–4XB0 (dt., engl., frz.)
Bestellort: WK Fürth

**/SYI/** SINUMERIK 840Di
**Systemüberblick** (Ausgabe 02.01)
Bestellnummer: 6FC5 298–6AE40–0AP0

## Hersteller–/Service–Dokumentation

### *a) Listen*

**/LIS/** SINUMERIK 840D/840Di/810D
SIMODRIVE 611D
**Listen** (Ausgabe 11.02)
Bestellnummer: 6FC5 297–6AB70–0AP3

### *b) Hardware*

**/BH/** SINUMERIK 840D/840Di/810D
**Bedienkomponenten–Handbuch** (HW) (Ausgabe 11.02)
Bestellnummer: 6FC5 297–6AA50–0AP2

**/BHA/** SIMODRIVE **Sensor**
**Absolutwertgeber mit Profibus–DP**
Benutzerhandbuch (HW) (Ausgabe 02.99)
Bestellnummer: 6SN1197–0AB10–0YP1

**/EMV/** SINUMERIK, SIROTEC, SIMODRIVE
**EMV–Aufbaurichtlinie**
Projektierungsanleitung (HW) (Ausgabe 06.99)
Bestellnummer: 6FC5 297–0AD30–0AP1

**/GHA/** **ADI4 – Analoge Antriebsschnittstelle für 4 Achsen** (Ausgabe 09.02)
Gerätehandbuch
Bestellnummer: 6FC5 297–0BA01–0AP0

**/PHC/** SINUMERIK 810D
**Handbuch Projektierung** (HW) (Ausgabe 11.02)
Bestellnummer: 6FC5 297–6AD10–0AP1

**/PHD/** SINUMERIK 840D
**Handbuch Projektierung NCU 561.2–573.4** (HW) (Ausgabe 10.02)
Bestellnummer: 6FC5 297–6AC10–0AP2

**/PMH/** SIMODRIVE **Sensor**
**Hohlwellenmesssystem SIMAG H**
Projektierungs–/Montageanleitung (HW) (Ausgabe 07.02)
Bestellnummer: 6SN1197–0AB30–0AP1

### *c) Software*

**/FB1/** SINUMERIK 840D/840Di/810D
**Funktionsbeschreibung Grundmaschine** (Teil 1) – (Ausgabe 11.02) –
(im folgenden sind die enthaltenen Bücher aufgeführt)
Bestellnummer: 6FC5 297–6AC20–0AP2

A2 Diverse Nahtstellensignale
A3 Achsüberwachungen, Schutzbereiche
B1 Bahnsteuerbetrieb, Genauhalt und Look Ahead
B2 Beschleunigung
D1 Diagnosehilfsmittel
D2 Dialogprogrammierung
F1 Fahren auf Festanschlag
G2 Geschwindigkeiten, Soll–/Istwertsysteme, Regelung
H2 Hilfsfunktionsausgabe an PLC
K1 BAG, Kanal, Programmbetrieb
K2 Achsen, Koordinatensysteme, Frames, Werkstücknahes Istwertsystem, Externe Nullpunktversch.
K4 Kommunikation
N2 NOT AUS
P1 Planachsen
P3 PLC–Grundprogramm
R1 Referenzpunktfahren
S1 Spindeln
V1 Vorschübe
W1 Werkzeugkorrektur

**/FB2/** SINUMERIK 840D/840Di/810D(CCU2)
**Funktionsbeschreibung Erweiterungsfunktionen** (Teil 2) –(Ausgabe 11.02)–
einschließlich FM–NC: Drehen, Schrittmotor
(im folgenden sind die enthaltenen Bücher aufgeführt)
Bestellnummer: 6FC5 297–6AC30–0AP2

A4 Digitale und analoge NCK–Peripherie
B3 Mehrere Bedientafeln und NCUs
B4 Bedienung über PG/PC
F3 Ferndiagnose
H1 Handfahren und Handradfahren
K3 Kompensationen

| | |
|---|---|
| K5 | BAGs, Kanäle, Achstausch |
| L1 | FM–NC lokaler Bus |
| M1 | Kinematische Transformation |
| M5 | Messen |
| N3 | Softwarenocken, Wegschaltsignale |
| N4 | Stanzen und Nibbeln |
| P2 | Positionierachsen |
| P5 | Pendeln |
| R2 | Rundachsen |
| S3 | Synchronspindel |
| S5 | Synchronaktionen (bis SW 3) |
| S6 | Schrittmotorsteuerung |
| S7 | Speicherkonfiguration |
| T1 | Teilungsachsen |
| W3 | Werkzeugwechsel |
| W4 | Schleifen |

**/FB3/** SINUMERIK 840D/840Di/810D(CCU2)
**Funktionsbeschreibung Sonderfunktionen** (Teil 3) – (Ausgabe 11.02) –
(im folgenden sind die enthaltenen Bücher aufgeführt)
Bestellnummer: 6FC5 297–6AC80–0AP2

| | |
|---|---|
| F2 | 3 bis 5–Achs–Transformation |
| G1 | Gantry–Achsen |
| G3 | Taktzeiten |
| K6 | Konturtunnelüberwachung |
| M3 | Achskopplungen und ESR |
| S8 | Konstante Werkstückdrehzahl für Centerless Schleifen |
| T3 | Tangentialsteuerung |
| TE0 | Installation und Aktivierung der Compilezyklen |
| TE1 | Abstandsregelung |
| TE2 | Analoge Achse |
| TE3 | Drehzahl–/Drehmomentkopplung, Master–Slave |
| TE4 | Transformationspaket Handling |
| TE5 | Sollwertumschaltung |
| TE6 | MKS–Kopplung |
| TE7 | Wiederaufsetzen – Retrace Support |
| TE8 | Taktunabhängige bahnsynchrone Schaltsignalausgabe |
| V2 | Vorverarbeitung |
| W5 | 3D–Werkzeugradiuskorrektur |

**/FBA/** SIMODRIVE 611D/SINUMERIK 840D/810D
Funktionsbeschreibung **Antriebsfunktionen** (Ausgabe 11.02)
(im folgenden sind die enthaltenen Kapitel aufgeführt)
Bestellnummer: 6SN1 197–0AA80–0AP9

| | |
|---|---|
| DB1 | Betriebsmeldungen/Alarmreaktionen |
| DD1 | Diagnosefunktionen |
| DD2 | Drehzahlregelkreis |
| DE1 | Erweiterte Antriebsfunktionen |
| DF1 | Freigaben |
| DG1 | Geberparametrieung |
| DL1 | MD des Linearmotors |
| DM1 | Motor–/Leistungsteilparameter und Reglerdaten berechnen |
| DS1 | Stromregelkreis |
| DÜ1 | Überwachungen/Begrenzungen |

**/FBAN/** SINUMERIK 840D/SIMODRIVE 611 digital
Funktionsbeschreibung
**ANA–Modul** (Ausgabe 02.00)
Bestellnummer: 6SN1 197–0AB80–0AP0

**/FBD/** SINUMERIK 840D
Funktionsbeschreibung **Digitalisieren** (Ausgabe 07.99)
Bestellnummer: 6FC5 297–4AC50–0AP0

DI1 Inbetriebnahme
DI2 Scan mit taktilem Sensor (scancad scan)
DI3 Scan mit Laser (scancad laser)
DI4 Fräsprogrammerstellung (scancad mill)

**/FBDN/** IT–Solutions
**NC–Datenverwaltung Server** (DNC NT–2000) (Ausgabe 01.02)
Funktionsbeschreibung
Bestellnummer: 6FC5 297–5AE50–0AP2

**/FBDT/** SINUMERIK 840D/840Di/810D
IT–Solutions
**SinDNC NC–Datenübertragung über Netzwerk** (Ausgabe 09.02)
Funktionsbeschreibung
Bestellnummer: 6FC5 297–5AE70–0AP0

**/FBFA/** SINUMERIK 840D/840Di/810D
Funktionsbeschreibung
**ISO–Dialekte für SINUMERIK** (Ausgabe 11.02)
Bestellnummer: 6FC5 297–6AE10–0AP3

**/FBFE/** SINUMERIK 840D/840Di/810D
Funktionsbeschreibung **Ferndiagnose** (Ausgabe 11.02)
Bestellnummer: 6FC5 297–0AF00–0AP2

**/FBH/** SINUMERIK 840D/840Di/810D
**HMI–Programmierpaket** (Ausgabe 11.02)
Bestellnummer: (ist Bestandteil der SW–Lieferung)

Teil 1 Benutzeranleitung
Teil 2 Funktionsbeschreibung

**/FBHLA/** SINUMERIK 840D/SIMODRIVE 611 digital
Funktionsbeschreibung **HLA–Modul** (Ausgabe 04.00)
Bestellnummer: 6SN1 197–0AB60–0AP2

**/FBMA/** SINUMERIK 840D/810D
Funktionsbeschreibung **ManualTurn** (Ausgabe 08.02)
Bestellnummer: 6FC5 297–6AD50–0AP0

**/FBO/** SINUMERIK 840D/810D
**Projektierung Bedienoberfläche OP 030** (Ausgabe 09.01)
Funktionsbeschreibung
Bestellnummer: 6FC5 297–6AC40–0AP0

| | |
|---|---|
| BA | Bedienanleitung |
| EU | Entwicklungsumgebung (Projektierpaket) |
| PS | nur Online: Projektiersyntax (Projektierpaket) |
| PSE | Einführung in die Projektierung der Bedienoberfläche |
| IK | Installationspaket: Softwareupdate und Konfiguration |

**/FBP/** SINUMERIK 840D
Funktionsbeschreibung **C–PLC–Programmierung** (Ausgabe 03.96)
Bestellnummer: 6FC5 297–3AB60–0AP0

**/FBR/** SINUMERIK 840D/810D
IT–Solutions
Funktionsbeschreibung **Rechnerkopplung** (SinCOM) (Ausgabe 09.01)
Bestellnummer: 6FC5 297–6AD60–0AP0

| | |
|---|---|
| NFL | Nahtstelle zum Fertigungsleitrechner |
| NPL | Nahtstelle zu PLC/NCK |

**/FBSI/** SINUMERIK 840D / SIMODRIVE 611 digital
Funktionsbeschreibung **SINUMERIK Safety Integrated** (Ausgabe 09.02)
Bestellnummer: 6FC5 297–6AB80–0AP1

**/FBSP/** SINUMERIK 840D/810D
Funktionsbeschreibung **ShopMill** (Ausgabe 11.02)
Bestellnummer: 6FC5 297–6AD80–0AP1

**/FBST/** SIMATIC (Ausgabe 01.01)
Funktionsbeschreibung **FM STEPDRIVE/SIMOSTEP**
Bestellnummer: 6SN1 197–0AA70–0YP4

**/FBSY/** SINUMERIK 840D/840Di/810D
Funktionsbeschreibung **Synchronaktionen** (Ausgabe10.02)
Bestellnummer: 6FC5 297–6AD40–0AP2

**/FBT/** SINUMERIK 840D/810D
Funktionsbeschreibung **ShopTurn** (Ausgabe 03.03)
Bestellnummer: 6FC5 297–6AD70–0AP2

**/FBTC/** SINUMERIK 840D/810D
IT–Solutions
**SINUMERIK Tool Data Communication SinTDC** (Ausgabe 01.02)
Funktionsbeschreibung
Bestellnummer: 6FC5 297–5AF30–0AP0

**/FBTD/**
SINUMERIK 840D/810D
IT–Solutions
**Werkzeugbedarfsermittlung** (SinTDI) mit Online–Hilfe (Ausgabe 02.01)
Funktionsbeschreibung
Bestellnummer: 6FC5 297–6AE00–0AP0

**/FBU/**
**SIMODRIVE 611 universal/universal E**
Regelungskomponente für Drehzahlregelung und Positionieren
Funktionsbeschreibung (Ausgabe 02.02)
Bestellnummer: 6SN1 197–0AB20–0AP5

**/FBW/**
SINUMERIK 840D/840Di/810D
Funktionsbeschreibung **Werkzeugverwaltung** (Ausgabe 10.02)
Bestellnummer: 6FC5 297–6AC60–0AP1

**/FBWI/**
SINUMERIK 840D/840Di/810D
Funktionsbeschreibung **WinTPM** (Ausgabe 02.02)
Bestellnummer: Dokument ist Bestandteil der Software

**/HBA/**
SINUMERIK 840D/840Di/810D
**Handbuch @Event** (Ausgabe 03.02)
Bestellnummer: 6AU1900–0CL20–0AA0

**/HBI/**
SINUMERIK 840Di
**Handbuch** (Ausgabe 09.02)
Bestellnummer: 6FC5 297–6AE60–0AP1

**/INC/**
SINUMERIK 840D/840Di/810D
Inbetriebnahme–Tool **SINUMERIK SinuCOM NC** (Ausgabe 02.02)
Systembeschreibung
Bestellnummer: (ist Bestandteil der Online–Hilfe des IBN–Tools)

**/PAP/**
SIMODRIVE **Sensor**
**Absolutwertgeber mit Profibus–DP**
Benutzeranleitung (Ausgabe 02.99)
Bestellnummer: 6SN1197–0AB10–0YP1

**/PFK/**
**SIMODRIVE**
Projektierungsanleitung **1FT5–/1FT6–/1FK6–Motoren** (Ausgabe 12.01)
Drehstrom–Servomotoren für Vorschub– und Hauptspindelantriebe
Bestellnummer: 6SN1 197–0AC20–0AP0

**/PJE/**

SINUMERIK 840D/810D
**Projektierpaket HMI Embedded** (Ausgabe 08.01)
Funktionsbeschreibung : Softwareupdate, Konfiguration, Installation
Bestellnummer: 6FC5 297–6EA10–0AP0
(die Schrift PS Projektiersyntax ist Bestandteil der SW–Lieferung und als pdf verfügbar)

**/PJFE/**

**SIMODRIVE**
Projektierungsanleitung **Synchron–Einbaumotoren 1FE1**
Drehstrommotoren für Hauptspindelantriebe (Ausgabe 09.01)
Bestellnummer: 6SN1 197–0AC00–0AP1

**/PJLM/**

**SIMODRIVE**
Projektierungsanleitung **Linearmotoren 1FN1, 1FN3** (Ausgabe 06.02)
ALL Allgemeines zum Linearmotor
1FN1 Drehstrom Linearmotor 1FN1
1FN3 Drehstrom Linearmotor 1FN3
CON Anschlußtechnik
Bestellnummer: 6SN1 197–0AB70–0AP4

**/PJM/**

**SIMODRIVE**
Projektierungsanleitung **Motoren** (Ausgabe 11.00)
Drehstrommotoren für Vorschub– und Hauptspindelantriebe
Bestellnummer: 6SN1 197–0AA20–0AP5

**/PJTM/**

**SIMODRIVE**
Projektierungsanleitung **Einbau–Torquemotoren 1FW6** (Ausgabe 08.02)
Bestellnummer: 6SN1 197–0AD00–0AP0

**/PJU/**

**SIMODRIVE 611**
Projektierungsanleitung **Umrichter** (Ausgabe 05.01)
Bestellnummer: 6SN1 197–0AA00–0AP5

**/PMS/**

**SIMODRIVE** (Ausgabe 04.02)
Projektierungsanleitung **ECO–Motorspindel** für Hauptspindelantriebe
Bestellnummer: 6SN1 197–0AD04–0AP0

**/POS1/**

**SIMODRIVE POSMO A** (Ausgabe 08.02)
Dezentraler Positioniermotor am PROFIBUS DP, Benutzerhandbuch
Bestellnummer: 6SN2197–0AA00–0AP3

**/POS2/**

**SIMODRIVE POSMO A**
Montageanleitung (liegt jedem POSMO A bei)

**/POS3/**

**SIMODRIVE POSMO SI/CD/CA** (Ausgabe 08.02)
Dezentrale Servo Antriebstechnik, Benutzerhandbuch
Bestellnummer: 6SN2197–0AA20–0AP3

**/PPH/**
**SIMODRIVE**
Projektierungsanleitung **1PH2–/1PH4–/1PH7–Motoren** (Ausgabe 12.01)
Drehstrom–Asynchronmotoren für Hauptspindelantriebe
Bestellnummer: 6SN1 197–0AC60–0AP0

**/PPM/**
**SIMODRIVE**
Projektierungsanleitung **Hohlwellenmotoren** (Ausgabe 10.01)
Hohlwellenmotoren für Hauptspindelantriebe 1PM4 und 1PM6
Bestellnummer: 6SN1 197–0AD03–0AP0

**/S7H/**
**SIMATIC S7–300** (Ausgabe 2002)
– Referenzhandbuch: CPU–Daten (HW–Beschreibung)
– Referenzhandbuch: Baugruppendaten
– Handbuch technonlogische Funktionen
– Installationshandbuch
Bestellnummer: 6ES7 398–8FA10–8AA0

**/S7HT/**
**SIMATIC S7–300** (Ausgabe 03.97)
Handbuch: STEP 7, Grundwissen, V. 3.1
Bestellnummer: 6ES7 810–4CA02–8AA0

**/S7HR/**
**SIMATIC S7–300** (Ausgabe 03.97)
Handbuch: STEP 7, Referenzhandbücher, V. 3.1
Bestellnummer: 6ES7 810–4CA02–8AR0

**/S7S/**
**SIMATIC S7–300** (Ausgabe 04.97)
Positionierbaugruppe **FM 353** für Schrittantrieb
Bestellung zusammen mit dem Projektierpaket

**/S7L/**
**SIMATIC S7–300** (Ausgabe 04.97)
Positionierbaugruppe **FM 354** für Servoantrieb
Bestellung zusammen mit dem Projektierpaket

**/S7M/**
**SIMATIC S7–300** (Ausgabe 01.01)
Mehrachsbaugruppe **FM 357.2** für Servo– bzw. Schrittantrieb
Bestellung zusammen mit dem Projektierpaket

**/SP/**
**SIMODRIVE 611–A/611–D,**
**SimoPro 3.1**
Programm zur Projektierung von Werkzeugmaschinen–Antrieben
Bestellnummer: 6SC6 111–6PC00–0AA□, Bestellort: WK Fürth

### d) Inbetriebnahme

**/IAA/** **SIMODRIVE 611A**
**Inbetriebnahmeanleitung** (Ausgabe 10.00)
Bestellnummer: 6SN 1197–0AA60–0AP6

**/IAC/** SINUMERIK 810D
**Inbetriebnahmeanleitung** (Ausgabe 03.02)
(einschl. Beschreibung der Inbetriebnahme–Software
SIMODRIVE 611D)
Bestellnummer: 6FC5 297–6AD20–0AP0

**/IAD/** SINUMERIK 840D/SIMODRIVE 611 digital
**Inbetriebnahmeanleitung** (Ausgabe 11.02)
(einschl. Beschreibung der Inbetriebnahme–Software
SIMODRIVE 611 digital)
Bestellnummer: 6FC5 297–6AB10–0AP2

**/IAM/** SINUMERIK 840D/840Di/810D
**Inbetriebnahmeanleitung HMI/MMC** (Ausgabe 11.02)
Bestellnummer: 6FC5 297–6AE20–0AP2

AE1 Aktualisierungen/Ergänzungen
BE1 Bedienoberfläche ergänzen
HE1 Online–Hilfe
IM2 Inbetriebnahme HMI Embedded
IM4 Inbetriebnahme HMI Advanced
TX1 Fremdsprachentexte erstellen

# Index

## Zeichen



## A



## B



## C



## D



## E



## F



## G



## I



## K



## L



## M

## N



## O



## P



## R



## S



## T

## U



## V



## W

# Platz für Notizen

| An<br>SIEMENS AG<br>A&D MC BMS<br>Postfach 3180<br><br>D–91050 Erlangen<br><br>(Tel. 0180 / 5050 – 222 [Hotline]<br>Fax 09131 / 98 – 2176 [Dokumentation]<br>E–Mail: motioncontrol.docu@erlf.siemens.de) | **Vorschläge**<br><br>**Korrekturen**<br>für Druckschrift:<br><br>SINUMERIK 840D/840Di/810D<br>Funktionsbeschreibung<br>Synchronaktionen<br><br>Hersteller–Dokumentation |
|---|---|
| **Absender**<br><br>Name<br>Anschrift Ihrer Firma/Dienststelle<br>Straße<br>PLZ: Ort:<br>Telefon: /<br>Telefax: / | Funktionsbeschreibung<br><br>Bestell–Nr.: 6FC5 297–6AD40–0AP2<br>Ausgabe: 11.02<br><br>Sollten Sie beim Lesen dieser Unterlage auf Druckfehler gestoßen sein, bitten wir Sie, uns diese mit diesem Vordruck mitzuteilen. Ebenso dankbar sind wir für Anregungen und Verbesserungsvorschläge. |

**Vorschläge und/oder Korrekturen**

# Dokumentationsübersicht SINUMERIK 840D/840Di/810D (11.2002)

## Allgemeine Dokumentation

- *SINUMERIK* 840D/810D — Werbeschrift
- *SINUMERIK* 840D/840Di/ 810D/ — Katalog Bestellu. NC 60 ***)**
- *SINUMERIK SIROTEC SIMODRIVE* Zubehör — Katalog Zubehör NC–Z

## Anwender–Dokumentation

- *SINUMERIK* 840D/810D/ FM–NC — **AutoTurn** – Kurzanleitung – Programmieren /Einrichten
- *SINUMERIK* 840D/840Di 810D — Bedienungsanl. **– HT 6**
- *SINUMERIK* 840D/840Di/ 810D — **Diagnose–anleitung *)**
- *SINUMERIK* 840D/840Di/ 810D — **Bedienungsanl. *)** – Kurzanleitung – HMI Embedded – HMI Advanced

## Anwender–Dokumentation

- *SINUMERIK* 840D/840Di/ 810D — **Programmieranl.** – Kurzanleitung – Grundlagen ***)** – Arbeitsvorbereit. ***)** – Zyklen – Meßzyklen – ISO Turning/Milling
- *SINUMERIK* 840D/810D — Bedienungsanl. – **ManualTurn** – Kurzanl. ManualTurn – **ShopMill** – Kurzanl. ShopMill – **ShopTurn** – Kurzanl. ShopTurn
- *SINUMERIK* 840Di — **Systemüberblick**

## Hersteller–/Service–Dokumentation

- *SINUMERIK* — **Projektierung** (HW) ***)** – 810D – 840D
- *SINUMERIK* 840D/840Di/ 810D — **Bedien–komponenten** (HW) ***)**
- *SINUMERIK* 840D/810D — Funktionsbeschr. – **ManualTurn** – **ShopMill** – **ShopTurn**
- *SINUMERIK* 840D/840Di/ 810D — Funktionsbeschr. **Synchronaktionen**

## Hersteller–/Service–Dokumentation

- *SINUMERIK SIMODRIVE* 611D 840D/810D — Funktionsbeschr. **Antriebsfunktion *)**
- *SINUMERIK* 840D/840Di/ 810D — **Funktionsbeschr.** – Grundmaschine ***)** – Erweiterungsfunkt. – Sonderfunktionen
- *SINUMERIK* 840D/840Di/ 810D — Funktionsbeschr. **Werkzeug–verwaltung**
- *SINUMERIK* 840D/810D — **Projektier–Pkt.** HMI Embedded
- *SINUMERIK* 840D/810D — Funktionsbeschr. Projekt. Bedien–oberfläche **OP 030**
- *SINUMERIK* 840D/810D — **IT–Solutions** – Rechnerkopplung – Werkzeugbedarfse. – NC–Datenverwalt. – NC–Datenübertrag. – Tool Data Communica.

## Hersteller–/Service–Dokumentation

- *SINUMERIK SIMODRIVE* — Funktionsbeschr. SINUMERIK **Safety Integrated**
- *SINUMERIK* 840D — Funktionsbeschr. **Digitalisieren**
- *SINUMERIK SIMODRIVE* — **Inbetriebnameanl. *)** – 810D – 840D/611D – HMI
- *SINUMERIK SIMODRIVE* 840D/840Di 810D 611D — **Listen *)**
- *SINUMERIK SIMODRIVE* 840D 611D — Funktionsbeschr. **Linearmotor**
- *SINUMERIK SIMODRIVE* 840D 611D — Funktionsbeschr. **– Hydraulikmodul – Analogmodul**
- *SINUMERIK SIMODRIVE SIROTEC* — **EMV**–Richtlinien

## Elektronische Dokumentation

- *SINUMERIK SIMODRIVE* 840D/840Di/ 810D 611, Motoren — **DOC ON CD *)** Das SINUMERIK–System

## Hersteller–/Service–Dokumentation

- *SINUMERIK* 840D/840Di/ 810D — Funktionsbeschr. **ISO–Dialekte** für SINUMERIK
- *SINUMERIK* 840Di — **Handbuch** (HW + Inbetriebn.)
- *SINUMERIK* 840D/810D — Funktionsbeschr. **Ferndiagnose**
- *SINUMERIK* 840D/840Di/ 810D — **Handbuch @ Event**

---

***)** Empfohlener Minimalumfang der Dokumentation

**Siemens AG**
Automatisierungs- und Antriebstechnik
Motion Control Systems
Postfach 3180, D – 91050 Erlangen
Bundesrepublik Deutschland

www.ad.siemens.de


Bestell-Nr.: 6FC5297-6AD40-0AP2

Gedruckt in der Bundesrepublik Deutschland